MASTER EN SCIENCES INFORMATIQUES

Modélisation opérationnelle et intentionnelle de workflows: application aux processus de soins médicaux

Roucoux, François

Award date:
2009

Modélisation opérationnelle et intentionnelle de workflows : application aux processus de soins médicaux

François Roucoux

Mémoire présenté en vue de l'obtention du grade de Licencié en Informatique

Année académique 2008 – 2009

Faculté Universitaire Notre-Dame de la Paix, Namur
Institut d'Informatique

Résumé

Le monde médical adopte de plus en plus une démarche de gestion des soins orientée-processus. Pourtant, on trouve peu de systèmes de gestion de workflows déployés dans les hôpitaux pour supporter ces processus. La médecine nécessite à la fois des procédures bien définies et une grande flexibilité opérationnelle. Les systèmes actuels, purement procéduraux, semblent inadaptés. Parmi les nombreux formalismes de description de workflows, ce mémoire propose d'utiliser le calcul événementiel pour construire la spécification précise d'un processus de soins. Cette spécification contient à la fois un modèle intentionnel des objectifs et un modèle opérationnel des activités. Un mécanisme est proposé qui permet au modèle intentionnel d'influencer le déroulement du modèle opérationnel et d'améliorer ainsi sa flexibilité.

Mots-clés

processus de soins, workflows, modèle d'objectifs, calcul événementiel

Abstract

Care management is more and more process-oriented in the medical world. However few workflow management systems are effectively deployed in hospitals to support these processes. Medicine requires both well-defined procedures and operational flexibility. Current systems, purely procedural, seem inadequate. Among many formalisms available for workflow description, this master thesis proposes the use of event calculus to build a precise specification of a care process. This specification contains both an intentional model of goals and an operational model of activities. We propose a mechanism that allows the intentional model to influence the operational model, improving its flexibility.

Keywords

care process, workflows, goal model, event calculus

Nous remercions notre promoteur, Jean-Marie Jacquet pour la totale liberté qu'il nous a laissée dans le choix du sujet, ses remarques, ses conseils pertinents ainsi que ses encouragements pendant l'écriture de ce mémoire.

Table des matières

Chapitre 1

Introduction

Le point de départ de ce travail est une expérience acquise en aidant des travailleurs de la santé en milieu hospitalier à exprimer et à formaliser leurs processus de soins. C'est également le résultat des frustrations rencontrées lors de ces efforts : confronté à la complexité du problème, au manque de flexibilité des formalismes disponibles et à l'absence d'un support méthodologique clair.

1.1 Contexte

Je m'intéresse depuis quelques années aux processus de soins, plus dans une perspective médicale que réellement informatique ou organisationnelle. Cet intérêt m'a permis de participer à la conception des guidelines nationaux dans le traitement du cancer colorectal [38]. Il m'a également amené à aider les membres d'une équipe de soignants à mettre en place un processus de prise en charge de ce même cancer dans un grand hôpital. Ce qui m'a chaque fois frappé, c'est la complexité et l'étendue du processus de soins lui-même mais également la difficulté de le représenter de la façon la moins ambiguë possible tout en restant compréhensible des soignants.

A peu près à la même période, j'ai eu l'occasion de suivre un cours sur la méthodologie d'ingénierie de spécification KAOS par un de ses créateurs : le Professeur Axel van Lamsweerde. J'avais déjà eu un bref contact avec cette méthodologie lors de mes études d'informatique. Deux choses m'ont principalement interpelé dans ce cours : le niveau de précision qu'il était possible d'atteindre et le côté naturel et systématique du mécanisme de raffinement d'objectifs. La précision s'obtient malheureusement au prix d'un cortège de formules qui la rend opaque au commun des mortels. J'ai malheureusement oublié la plupart des formules pour ne conserver que l'idée du raffinement d'objectifs.

Il y a de cela six mois, j'ai été invité, une nouvelle fois, à participer à la mise en place d'un processus de soins. Il s'agit cette fois-ci d'un itinéraire clinique de prise en charge de patientes atteintes d'un cancer du sein (la notion d'itinéraire clinique sera expliquée dans ce texte). Ce projet baptisé APCOS (Agile Patient-centered Care Orchestration System) consiste à développer un outil de support de cet itinéraire en coordonnant l'activité des soignants sur le terrain. Une des exigence non-fonctionnelle du projet est une grande simplicité d'utilisation. J'ai intégré une équipe de trois autres informaticiens et nous avons pensé cet outil avant tout comme un gestionnaire de tâches individuelles. Chaque soignant y a accès, soit sur des postes de travail fixes, soit sur des bornes présentes dans les couloirs des services et, dans le futur, sur des organiseurs personnels de poche. L'outil est alimenté par un modèle du workflow de l'itinéraire clinique, pour l'instant en cours de formalisation.

En réalité, cet itinéraire clinique fonctionne déjà depuis plusieurs années dans l'hôpital. Il n'a jamais

été réellement formalisé à part quelques vagues organigrammes punaisés aux murs des services. Et pourtant, il tourne et les patientes bénéficient d'une excellente prise en charge. Cette situation n'est pourtant pas idéale, notamment parce que lorsqu'un problème survient (par exemple, une patiente ne se présente plus à la consultation), le temps de réaction de l'équipe multidisciplinaire est en général assez long. La communication entre les services impliqués (gynécologie, oncologie clinique, radiothérapie, médecine nucléaire, chirurgie, ...) n'est pas optimale ce qui génère parfois du stress et des conflits. Enfin, il demeure difficile de se faire une idée précise du fonctionnement réel de l'itinéraire sur le terrain, de ce qui marche bien et de ce qui nécessiterait une amélioration.

Ce dernier point est en partie adressé par notre outil qui mesure des indices de qualité et de performance relatifs au déroulement du processus et à l'état du patient. Très tôt dans notre analyse, il est apparu que cette notion d'indices à mesurer était intimement liée à celle d'objectifs à atteindre. Cette notion d'objectifs thérapeutiques est d'ailleurs fréquemment mentionnée dans la littérature consacrée aux itinéraires cliniques. Nous avons décidé d'intégrer cette notion d'objectif dans l'application, dans un premier temps pour documenter les étapes clés du processus.

1.2 Objectifs du mémoire

A ce stade, au moins deux questions intéressantes se posent si l'on considère un itinéraire clinique comme une succession d'activités et de procédures qui s'enchaînent pour atteindre certains objectifs :

- Comment représenter explicitement les objectifs au même titre que les activités dans un modèle de processus ?
- Comment s'y prendre pour que ce soit la satisfaction des objectifs qui guide et influence l'enchaînement des activités du processus ?

L'objectif principal de ce mémoire est de trouver une réponse, au moins partielle, à ces deux questions. En marge de celui-ci, je poursuis au moins deux autres objectifs plus personnels dont je souhaite également faire profiter le lecteur :

- me familiariser au domaine des workflows et à leur formalisation.
- explorer la possibilité de construire un modèle d'objectifs pendant les phases d'analyse d'un processus de soins et de s'appuyer sur ce modèle pour construire le workflow du processus.

Première partie

Etat de l'art

Chapitre 2

Les workflows

On utilise des workflows pour représenter les processus métiers. Un workflow est un modèle opérationnel qui décrit les activités, rôles, ressources et applications externes liées au processus métier représenté. Ce chapitre fait un tour d'horizon des principaux concepts de ce domaine.

2.1 Les processus métier

Un processus métier (business process) est un ensemble de procédures ou d'activités qui, dans le contexte d'une organisation, produit un service, un produit ou réalise un objectif particulier. On retrouve des processus métier dans tous les types d'organisations humaines. Ces processus sont souvent liés à l'activité principale de l'organisation tels les processus de fabrication dans l'industrie manufacturière ou des processus de soins dans le domaine de la santé. Ils sont parfois liés à des activités secondaires qui supportent l'activité principale tels des processus logistiques ou de gestion.

Lorsqu'une organisation décide de formaliser ses processus internes, ils sont souvent représentés sous une forme diagrammatique (des boîtes et des flèches). Il existe un nombre très important de ces formats visuels, pour la plupart non-formalisés. Le plus répandu ou le plus à la mode dans le monde de l'entreprise actuel est sans doute BPMN (Business Process Modeling Notation). La figure 2.1 est un exemple de diagramme BPMN.

On représente un processus métier à l'aide d'un modèle de processus encore appelé modèle de workflow.

2.2 Les workflows

On définit un workflow comme une collection d'activités, de données, d'agents et de ressources coordonnés et mis en jeu lors de la réalisation d'un processus déterminé. Un workflow contient également les relations et les dépendances qui existent entre ces différents éléments.

On décrit un modèle de workflow à l'aide d'un langage de description de workflow. Les langages de description de workflow trouvent leur utilité dans deux domaines principaux :

l'orchestration de services web : les langages de description de workflow y sont utilisés pour spécifier des contrats de collaboration entre des services distribués. Cet usage requiert de pouvoir

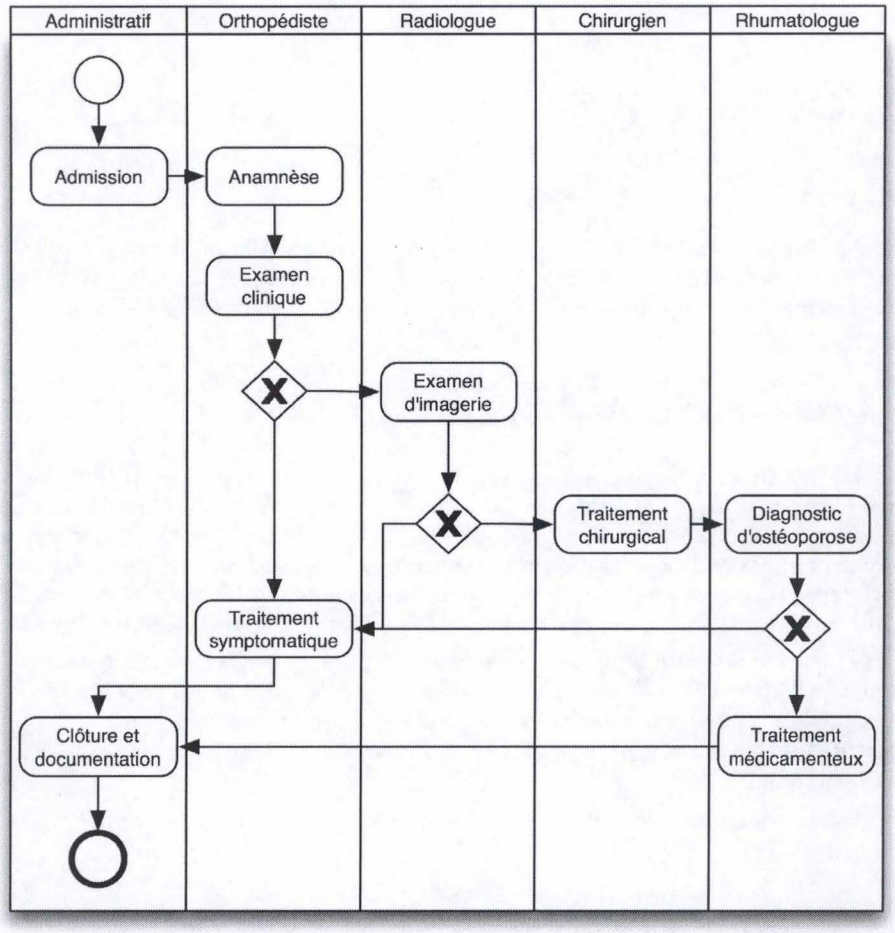

FIG. 2.1 – Figure : Notation BPMN de la prise en charge d'une fracture chez la personne âgée

exprimer les spécifications de processus dynamiques communiquants de façon précise et si possible concise. Certaines algèbres de processus et les langages qui en sont dérivés sont particulièrement bien adaptés à cette problématique (voir section 3.8).

la gestion et l'orchestration des processus métiers : dans ce domaine, les langages de description de workflow servent à définir des représentations informatiques de processus du monde réel. Ces représentations sont ensuite utilisées pour alimenter des systèmes de gestion de workflows qui supportent l'exécution des processus sur le terrain.

Dans ce travail, nous négligerons l'orchestration des services web pour nous intéresser exclusivement à la gestion de processus métiers, plus particulièrement dans le contexte des soins de santé.

2.2.1 Différentes perspectives sur les workflows

Un workflow peut se décrire selon différentes perspectives :

- le flux de contrôle
- les données
- les ressources et les agents
- d'autres aspects opérationnels

La perspective du flux de contrôle du workflow s'intéresse à sa décomposition en activités et à l'ordre d'exécution de ces activités. Cette perspective s'intéresse également aux contraintes temporelles implicites et explicites qui influencent la séquence d'exécution des activités.

La perspective des données concerne les données utilisées et produites par les différentes activités d'un workflow. L'existence de ces flots de données induit des pré et des post-conditions entre les activités. L'exécution de certaines activités dépend de données produites par d'autres activités. Par exemple, une activité qui consiste pour une infirmière à administrer un traitement dépend d'une autre activité dans laquelle un médecin a prescrit le traitement en question (le médicament concerné et sa dose). En général, la transmission des données d'une activité à l'autre est assurée par le système de gestion de workflow.

Une perspective particulière s'intéresse aux agents et aux ressources mis en jeux dans le workflow. Dans cette perspective, on retrouve la description des ressources et des contraintes de leur allocation aux différentes activités. On trouve également un descriptif des agents, de leurs compétences et responsabilités respectives quant à l'exécution des activités ainsi que d'éventuelles relations de hiérarchie qui existent entre eux.

Enfin, la perspective opérationnelle concerne les modes d'interactions du workflow avec son environnement : la façon d'accéder et d'invoquer des applications et des sources d'informations externes (dossier patient informatisé, agenda partagé, ...) ainsi que la façon d'interagir avec des êtres humains.

2.2.2 Les instances de workflow

On appelle instance de workflow ou cas, toute exécution d'un schéma de workflow. Par exemple, dans la situation ou un schéma de workflow décrit la prise en charge du cancer du sein, un cas est créé (instancié) pour chaque patiente qui se présente à l'hôpital avec ce type de pathologie. Donc, plusieurs cas issus d'un même schéma de workflow peuvent s'exécuter en même temps. Ces concepts de schéma et d'instance de workflow sont à rapprocher des concepts de classes et d'objets (instances de classes) dans les langages de programmation orientés-objet.

2.2.3 Les activités

Les activités sont les composants principaux d'un workflow. Elles correspondent aux étapes individuelles du processus métier modélisé et représentent un certain travail à effectuer. Des agents humains ou logiciels sont responsables de l'exécution de ces activités. C'est la raison pour laquelle on distingue les activités manuelles (exécutées par des humains) des activités automatiques (exécutées par des ordinateurs ou d'autres automates assimilés).

Les activités sont caractérisées entre autre par leur durée qui peut-être fixe ou variable. La plupart des activités automatiques ont une durée fixe. Les activités réalisées par les êtres humains sont le plus souvent de durée variable.

Du point de vue d'une application de gestion de workflow, une activité peut être dans différents états d'exécution (figure 2.2). Ce sont des événements externes (start, end, stop) qui font passer l'activité d'un état à l'autre.

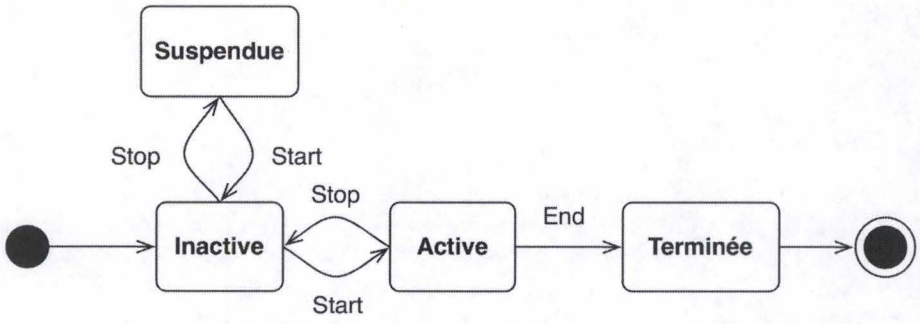

FIG. 2.2 – Etats d'exécution possibles pour une activité (adapté du modèle de référence de la WfMC [34])

Ce n'est pas le système de gestion de workflow qui est responsable de l'exécution des activités. Les activités manuelles à réaliser sont instanciées dans des listes de tâches ("worklists" ou "tasklists") attribuées à des agents humains. La figure 2.3 présente une capture d'écran d'un tel gestionnaire de tâches. Les activités automatiques correspondent à l'invocation d'applications connexes par le système de gestion de workflow (figure 2.5).

Dans un modèle de workflow, ce sont les relations de dépendance qui existent entre les activités (flux de contrôle et de données, contraintes d'utilisation des ressources et de disponibilité des agents) qui déterminent la séquence d'exécution de ces activités.

2.2.4 Les agents et les rôles

Chaque activité d'un workflow est prise en charge par un agent. Le concept d'agent fait référence à un acteur ou à un groupe d'acteurs humains ou informatiques qui prennent part à l'exécution du workflow.

Plusieurs agents différents peuvent être qualifiés pour exécuter une activité donnée. C'est la raison d'être du concept de rôle. Un rôle est défini par un ensemble de responsabilités et un ensemble de compétences. Les responsabilités sont des objectifs assignés aux agents qui endossent ce rôle. Les

FIG. 2.3 – Gestionnaire individuel de tâches (application APCOS, en cours d'élaboration pour le support des itinéraires cliniques)

compétences sont des activités ou des groupes d'activités coordonnées (que nous appellerons procédures) que l'agent qui endosse le rôle est capable de mener à bien.

La notion de rôle est assez polymorphe. Elle peut correspondre à :

- une profession : médecin, infirmière, physicien médical ; ou à une capacité particulière
- un titre : coordinateur d'itinéraire clinique, responsable qualité, ...
- une spécialité particulière : spécialiste en chirurgie métastatique hépatique, infirmier praticien en circulation extra-corporelle, rythmologue, ...

Ces notions de rôle et de responsabilités sont très importantes dans le contexte médical en liaison avec des notions médico-légales et déontologiques.

2.3 Les systèmes de gestion de workflow

La gestion des workflows ea pour objectif d'offrir un support à la mise en oeuvre et au bon fonctionnement des processus métiers au sein d'une organisation. Un système de gestion de workflow (Workflow Management Systems) ou SGW est un système d'information générique qui offre du support à la modélisation, la vérification, l'exécution, la gestion et le monitoring de workflows.

Les organisations humaines utilisent des SGWs pour orchestrer et rationaliser leurs processus internes dont le bon déroulement dépend de ressources humaines et de systèmes d'information. Les SGWs contribuent à améliorer le fonctionnement de ces processus en fournissant aux acteurs humains et informatiques la bonne information au bon moment pour effectuer une activité spécifique. Pour ce faire, ils s'intègrent à d'autres technologies telles que la téléphonie, les applications web, les web services et l'e-mail.

Un tel système est générique car à priori utilisable dans n'importe quelle organisation de n'importe quel domaine d'activité qui fait appel à la notion de workflow. Nous verrons, plus tard dans ce travail que cette assertion doit être prise avec réserve car certains domaines, dont le domaine médical, posent des défis particuliers à l'implantation d'un SGW.

Un SGW fonctionne à partir de spécifications (modèles ou schémas) de workflows qui décrivent, le plus fidèlement possible, les activités d'un processus métier d'une organisation.

Les SGWs comportent différents outils de support aux différentes étapes du cycle de vie des processus (figure 2.4) :

- Des outils d'inférence de modèle de processus à partir de traces d'exécution laissée dans des systèmes d'informations
- Des outils de support à la modélisation et à l'analyse (outils de simulation ou de vérification statique) de processus
- Des outils de coordination de l'exécution des activités du processus sur le terrain en fonction d'un modèle
- Des outils de monitoring en temps réel ou en différé de l'exécution du processus. Ces outils peuvent produire des statistiques de performance ou de consommation de ressources

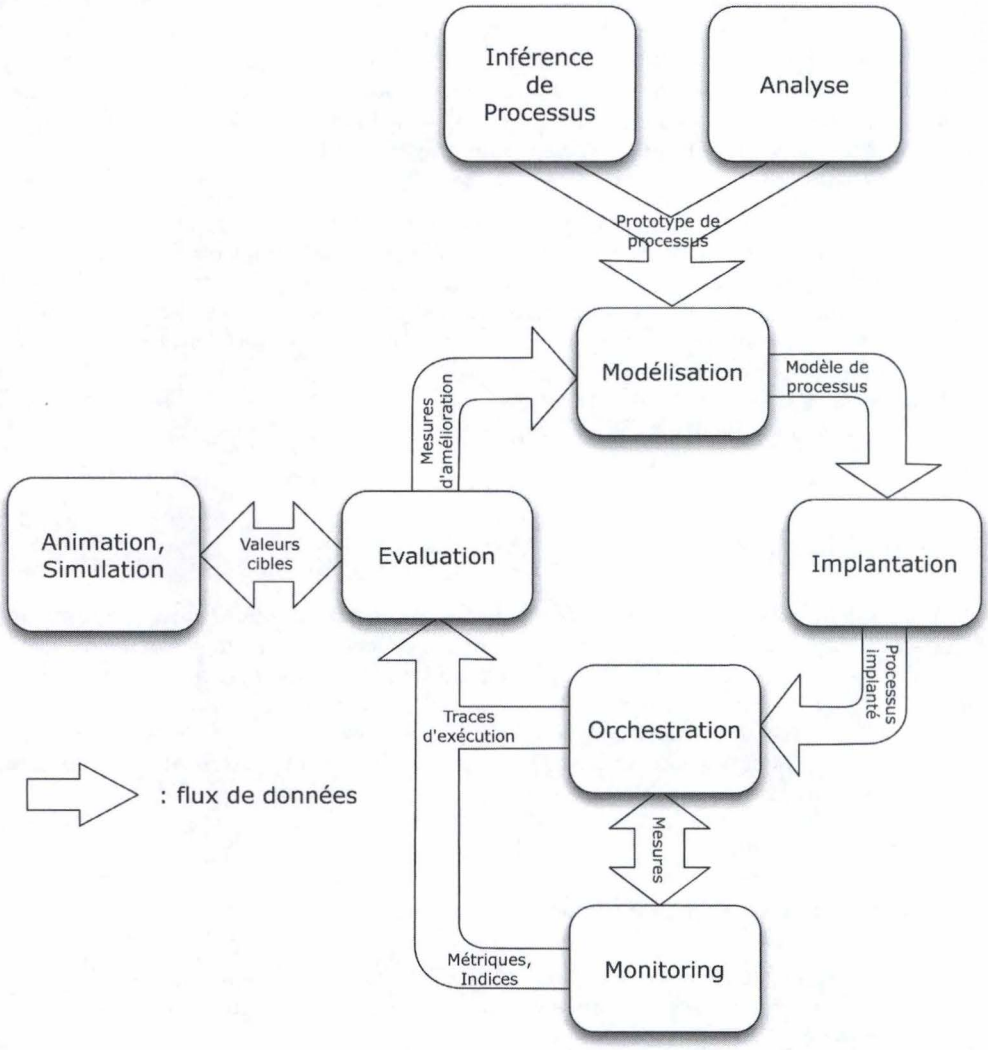

FIG. 2.4 – Les étapes du cycle de vie d'un workflow (adapté de la page 86 de l'ouvrage de Muehlen [52])

2.4 Le cycle de vie d'un workflow

2.4.1 L'analyse

La première étape dans la mise en place d'un système de gestion de workflow correspond à l'analyse des procédures existantes sur le terrain (figure 2.4). De par la complexité intrinsèque des processus médicaux, cette étape d'analyse nécessite un investissement important pour élucider le fonctionnement réel du processus. Cette phase correspond généralement à de nombreuses réunions et ateliers

impliquant des gestionnaires mais surtout les acteurs de terrain qui doivent expliquer la façon dont ils procèdent pour mener à bien le processus. Cette étape nécessite également une communication importante entre les groupes d'acteurs des différentes disciplines qui ne sont pas nécessairement au courant des activités menées et des objectifs poursuivis par les acteurs des autres groupes. Le produit final de cette phase d'analyse est un document qui reflète le plus fidèlement possible le processus actuel. Ce document comporte généralement une vision diagrammatique du processus qui représente la séquence des tâches, les acteurs impliqués, les canaux de communication nécessaires, ...

Des techniques de découverte (inférence) de processus [76] peuvent être appliquées très tôt dans la phase d'analyse. Ces techniques permettent de construire semi-automatiquement des versions très préliminaires de modèles de processus. Ces modèles qui reflètent les processus tels qu'exécutés sur le terrain serviront de base au travail d'analyse. Dans une phase plus avancée, des techniques de vérification de conformité pourront-être appliquées sur les modèles de processus qui résultent de l'analyse. L'apport de ces techniques de process mining permet d'accélérer considérablement la phase d'analyse traditionnellement importante consommatrice de ressources temporelles (plusieurs mois à plus d'une année pour les processus médicaux les plus complexes).

2.4.2 La modélisation

Une fois le processus analysé et le résultat de cette analyse documenté, il faut produire un modèle informatique exécutable de ce processus qui servira de base à l'orchestration des acteurs sur le terrain. La production de ce modèle nécessite un outil particulier : le modeleur. Une des propositions de ce travail est que ce modèle consiste en une combinaison d'informations procédurales et intentionnelles exécutables. En effet, notre postulat est que l'élicitation d'objectifs et leur raffinement en sous-objectifs mènent naturellement à la définition des procédures à exécuter pour les atteindre, d'agents impliqués, et de ressources nécessaires.

2.4.3 L'orchestration sur le terrain

Les modèles déployés seront « exécutés » par un outil d'orchestration qui assurera la collection de données sur les processus en cours ainsi que la planification des tâches et la coordination des différents acteurs qui les réalisent.

La métaphore bien connue de l'orchestre explique bien ce mécanisme : un chef d'orchestre coordonne les musiciens dans l'exécution d'un morceau. Diriger les musiciens consiste à leur envoyer des signaux gestuels pour que leurs prestations individuelles se déroulent de façon coordonnée et que le résultat d'ensemble soit le plus harmonieux possible. Pour l'aider dans sa direction, le chef d'orchestre consulte une partition qui lui indique quelles notes chaque musicien doit jouer à quel moment. Cette prestation peut avoir lieu à plusieurs reprises lors de différents concerts. L'orchestration informatique d'un processus suit le même principe : un logiciel orchestrateur ou moteur de workflow (le chef d'orchestre) interprète un modèle exécutable de processus (la partition) pour coordonner les acteurs de terrains (les musiciens) qui exécutent les tâches (émettre les notes avec leurs instruments) prévues par le modèle de processus. Pour coordonner les acteurs, l'orchestrateur leur envoie des signaux (liste de tâches à réaliser, rappels, avertissements, rapports ...). Le même modèle de processus peut être exécuté en parallèle pour des patients différents. L'orchestrateur est alors responsable de la bonne répartition du temps de travail des acteurs entre ces différentes instances de processus.

2.4.4 Le monitoring et l'évaluation

Pendant l'exécution des différentes instances d'un même processus, l'orchestrateur enregistre une série de paramètres : durée des tâches, coût, temps d'attentes, dépassement d'échéance, ... A partir de ces paramètres, il peut calculer en temps réel une série d'indicateurs de performance et les afficher sous la forme d'un tableau de bord du processus. Cette évaluation en temps réel est appelée monitoring. Elle constitue un retour utile et immédiat pour les gestionnaires du processus. Régulièrement, le système collecte l'ensemble des traces d'exécutions des processus sur une période de temps donnée. Cette somme d'informations fait l'objet d'une analyse rétrospective plus fouillée (évaluation de la performance cumulée, détection de non-conformités, inférence des échanges d'informations implicites entre acteurs, ...). Cette étape d'analyse met généralement en évidence les défauts de la version actuelle du modèle de processus. Le modèle est alors corrigé avant d'être utilisé pour un nouveau cycle d'orchestration. Cette succession d'analyses et de corrections successives place le modèle de processus dans un cycle d'amélioration continue.

2.5 Vers un modèle de référence

Le concept de workflow est tellement répandu qu'il existe un consortium international chargé de définir des standards d'interopérabilité entre les systèmes de gestion de workflows. Ce consortium, la WfMC (Workflow Management Coalition) regroupe plus de trois cents sociétés et organisations (dont IBM et Adobe) qui offrent des services ou qui contribuent à ce domaine.

On peut citer, parmi les principales réalisations de ce consortium :

- la spécification d'un modèle de référence pour les systèmes de gestions de workflow [34]. Ce modèle décrit de façon générique les principaux services offerts par ces systèmes, des propositions architecturales quant à leur implémentation, un modèle de référence pour les workflows eux-mêmes et un modèle d'interopérabilité entre ces systèmes.
- la spécification d'un langage de description de workflow basé sur XML et baptisé XPDL (XML process Definition Language). Ce langage est destiné au stockage et à l'échange de modèles de workflows aussi bien sous une forme textuelle que diagrammatique.
- la spécification du protocole d'interopérabilité Wf-XML qui permet à un système de gestion de workflow d'invoquer un processus sur un autre SGW et d'être tenu au courant de l'état de ce processus.
- un glossaire [99] bien utile dans ce domaine peu formalisé et en constante évolution.

2.6 Les motifs de workflows

A l'heure actuelle, la plupart des langages de description de workflows supportent un ensemble de structures de contrôle de base : la séquence, l'exécution conditionnelle, l'exécution parallèle et l'itération. Néanmoins, la sémantique opérationnelle de ces constructions pourtant simples varie d'un langage à l'autre. Ces différences se marquent dans des cas d'utilisation limites, lorsque ces constructions sont combinées pour représenter des comportements complexes. Les motifs de workflow (Workflow Patterns) constituent une tentative de description systématisée des principaux schémas d'agencement des éléments d'un workflow. Ils mettent en lumière certains de ces cas d'utilisation limite.

Ce concept de motif de workflow, dérivé des patterns de design en programmation, a été introduit entre 2000 et 2003 par W. M. P. van der Aalst et son équipe sous la forme de deux articles fondateurs [88, 90].

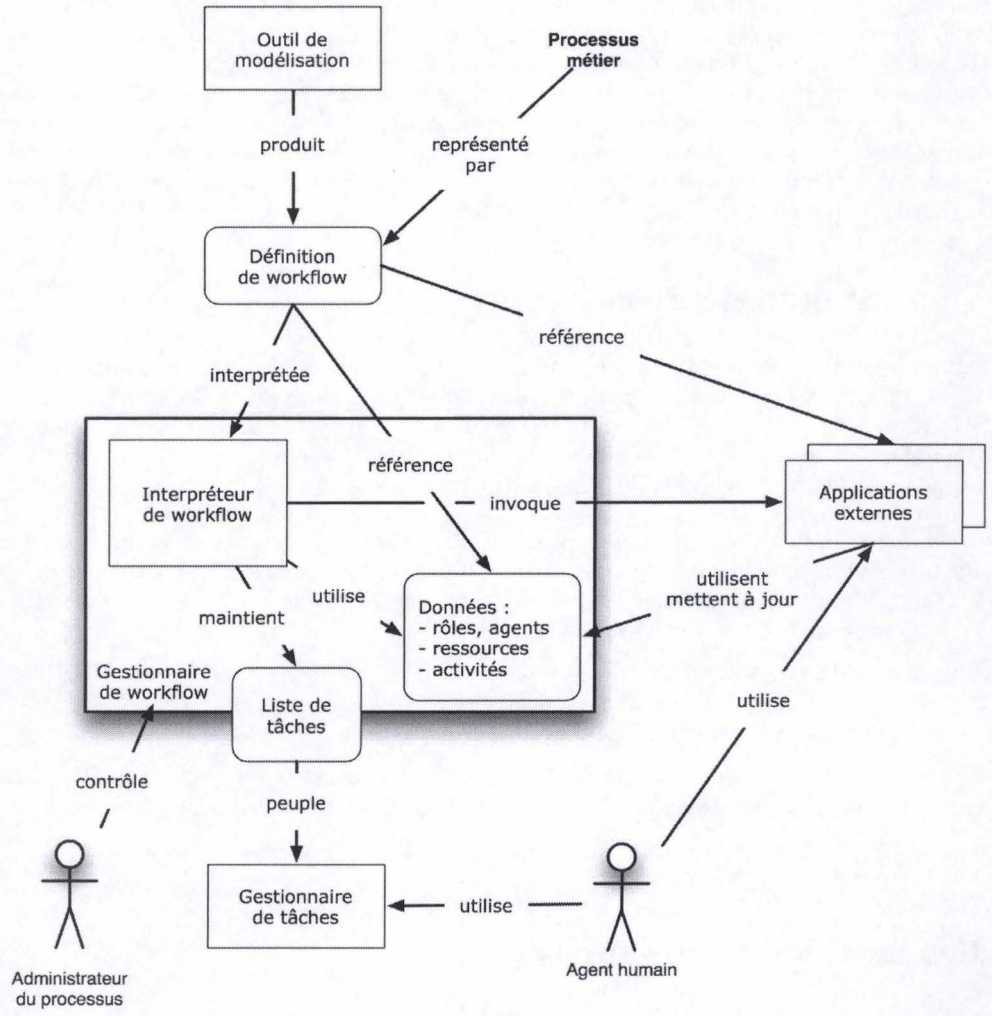

FIG. 2.5 – Synthèse des principaux concepts du domaine de la représentation et de l'orchestration des processus métiers (adapté du modèle de référence de la WfMC [34])

Ces deux articles proposent ensemble 20 motifs qui abordent les workflows essentiellement dans la perspective de leur flux de contrôle. Depuis, ce travail a été entendu par la même équipe pour proposer plus de 120 motifs classés selon 4 pespectives dont 3 ont été évoquées à la section 2.2.1 :

- les flux de contrôle
- les données
- les ressources
- les comportements exceptionnels

Pour rassembler ces motifs, les chercheurs se sont basés sur un dénominateur commun de fonctionnalités offertes par les SGWs contemporains de leur publication. Ils ont également tenu compte des exigences fonctionnelles les plus fréquemments exprimées par les utilisateurs de ces systèmes. La liste proposée n'est pas définitive et n'a aucune prétention d'exhaustivité. On peut la considérer comme une forme de benchmark destiné à évaluer l'expressivité (facilité de modélisation) et les fonctionnalités des langages de représentation de workflows et des SGWs. Dans cette optique, les systèmes ou les langages capables d'implémenter un maximum de motifs (score quantitatif) le plus simplement possible (analyse qualitative) sont considérés comme les meilleurs.

Ces motifs sont exprimés d'une façon indépendante de toute technologie. Ils ont été implémentés dans un nombre important de systèmes ce qui fournit une variété surprenante de solutions de représentation pour des structures de contrôle assez communes [19].

La suite de cette section présente brièvement les 20 motifs de contrôle originaux [90]. Une représentation visuelle des motifs de base sous la forme de diagrammes d'activités UML est également proposée. Les diagrammes d'activités ont été choisis car leur aspect est proche des organigrammes informels utilisés pour représenter des processus dans des organisations qui ont un faible niveau de formalisme. Les hôpitaux, que l'auteur de ce travail connaît bien, sont un exemple représentatif d'organisations aux processus peu formalisés.

Le lecteur intéressé par la représentation du jeu complet de motifs dans différentes notations graphiques, particulièrement les diagrammes d'activités, peut consulter :

- White [100] qui compare la représentation des 20 motifs initiaux de l'équipe de van der Aalst en BPMN et en diagrammes d'activités.
- Wohed [103] qui compare la représentation des mêmes motifs en YAWL (voir section 3.3) et en diagrammes d'activités.

Il est intéressant d'observer que dans ces deux publications, les représentations sous forme de diagrammes d'activités diffèrent de temps en temps. Ceci est sans doute lié à l'ambiguïté sémantique des diagrammes d'activité définis dans UML.

Pour comprendre la sémantique des motifs présentés, il est parfois fait référence à des jetons qui transitent d'une activité à l'autre au fur et à mesure de leur exécution. Lorsqu'un jeton est transféré dans une activité, cette activité passe dans l'état "active" c'est à dire prête à être exécutée et à passer ensuite dans l'état "terminée" une fois exécutée sur le terrain.

Les 20 motifs de workflows initiaux sont divisés en plusieurs groupes :

- les motifs de base
- les motifs de branchement avancés
- les motifs structurels

- les motifs d'instances multiples
- les motifs basés sur des états
- les motifs d'annulation

2.6.1 Les motifs de base

Les motifs de base capturent les flux de contrôle élémentaires des workflows. Ils constituent en quelque sorte les briques de base de la modélisation du flux de contrôle. Il suffisent à représenter la plupart des cas concrets. Les autres motifs illustrent la variété des problèmes potentiels de représentation.

Le motif *séquence* est défini comme une série ordonnée d'activités dont chaque activité passe dans l'état "active" après que la précédente soit passée dans l'état "terminée" (figure 2.2). La Workflow Management Coalition (WfMC) appelle ce comportement "routage séquentiel". Ce motif est représenté sous forme de diagramme d'activité comme une séquence d'activités connectées par un flux de contrôle. La direction des flèches détermine l'ordre de la séquence des activités. Lorsqu'une activité passe dans l'état "terminée", elle émet un jeton qui se déplace à travers le flux de contrôle vers l'activité suivante et la fait passer dans l'état "activée".

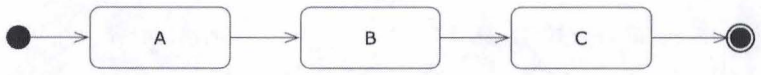

FIG. 2.6 – Motif de workflow n°1 : la séquence

Le motif *division parallèle* exprime une division du flux de contrôle en deux ou plusieurs flux parallèles qui permettent à deux ou plusieurs activités de démarrer en même temps. La WfMC appelle ce comportement *division-ET* (AND-split). Les diagrammes d'activités utilisent un noeud de bifurcation (fork node) pour créer un ensemble de chemins parallèles. Un jeton est généré pour chacun des flux de contrôle sortant de la bifurcation. Ils sont transférés aux activités qui suivent la bifurcation et qui deviennent potentiellement exécutables (activées) en parallèle.

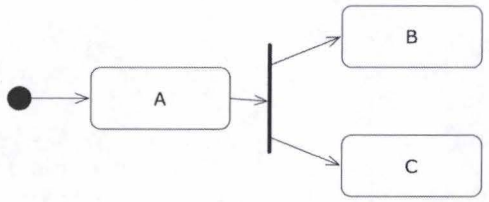

FIG. 2.7 – Motif de workflow n°2 : la division parallèle

Le motif *synchronisation parallèle* recombine les chemins générés par un motif "division parallèle". L'aspect de synchronisation exprime le fait que toutes les activités parallèles doivent être terminées avant de passer à l'activité suivante. La WfMC appelle ce comportement jonction-ET (AND-join). Les diagrammes d'activités utilisent un noeud de jonction (join node) pour recombiner un ensemble de flux

de contrôle parallèles. Un jeton issu de chaque flux incident doit parvenir au noeud de jonction pour qu'un jeton unique soit ré-émis vers l'activité suivante.

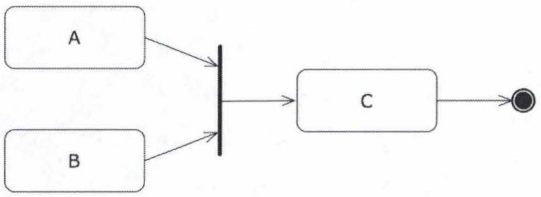

FIG. 2.8 – Motif de workflow n°3 : la synchronisation parallèle

Le motif *choix exclusif* permet de suivre un flux d'exécution parmi d'autres sur base de la valeur de vérité d'expressions conditionnelles ou gardes sur les flux. La WfMC appelle ce comportement division-OU exclusif (XOR-split). Les diagrammes d'activités utilisent un noeud décisionnel pour créer un ensemble de flux alternatifs exclusifs. Les flux de contrôle qui sortent du noeud décisionnel ont une expression conditionnelle attachée dont la valeur doit être vraie pour que le flux soit suivi. Une seule des expressions doit être vraie lors de leur évaluation. Il s'agit d'une propriété d'exclusivité des gardes qui doit être vérifiée pour que le workflow soit valide. Quand un jeton arrive sur le noeud décisionnel, les expressions sont évaluées dans un ordre non-déterministe. Le jeton est transmis au flux de contrôle dont la garde est vraie.

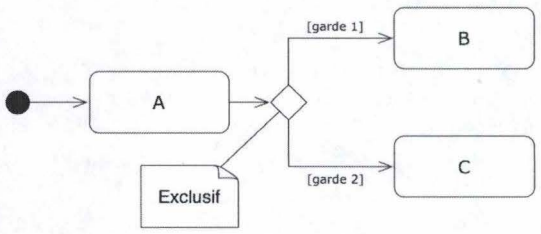

FIG. 2.9 – Motif de workflow n°4 : le choix exclusif

Le motif *synchronisation exclusive* représente la convergence de deux ou plusieurs flux de contrôle en un seul. L'appellation "synchronisation" du motif est inadéquate. Lorsqu'un jeton arrive par un des flux de contrôle, il est directement ré-émis vers l'activité suivante sans attendre l'arrivée d'autres jetons. Il ne s'agit donc pas d'une véritable synchronisation. La WfMC appelle ce comportement jonction-OU exclusif (XOR-join). Les diagrammes d'activités utilisent un noeud de fusion (merge node) pour faire converger un ensemble de chemins alternatifs. Quant un jeton arrive à ce noeud, il continue immédiatement son chemin vers le noeud suivant en suivant le flux de contrôle.

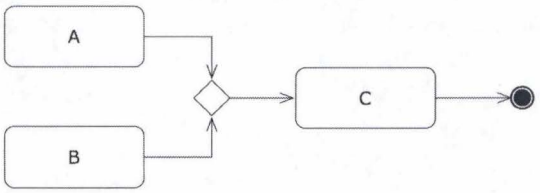

FIG. 2.10 – Motif de workflow n°5 : la synchronisation exclusive

2.6.2 Les motifs de branchement avancés

Les motifs de branchement avancés correspondent à des scénarios de contrôle plus sophistiqués qui ne sont d'ailleurs pas supportés par tous les SGWs et les langages de description de workflow.

Le motif *choix multiple* permet la sélection d'un ou plusieurs flux de contrôle en fonction de la valeur de vérité des gardes apposées sur ces flux. Dans ce motif, la propriété d'exclusivité des gardes n'est plus d'application. La situation dans laquelle aucune des gardes n'est vraie est cependant considérée comme invalide. En effet, dans ce cas, le workflow s'interromprait subitement au milieu de nulle part. La WfMC appelle ce comportement division-OU (OR-split).

Un motif "choix multiple" peut-être suivi des 4 motifs de jonction suivants :

- la fusion de synchronisation (synchronising merge)
- le discriminateur (discriminator)
- la multi-fusion (multi-merge)
- la jonction de sélection de n parmi m (n out of m join)

Le motif *fusion de synchronisation* attend tous les jetons issus des branches de la division-OU avant de ré-émettre un unique jeton.

Le motif *multi-fusion* représente un point du workflow dans lequel plusieurs flux de contrôle fusionnent sans contrôle particulier sur le flux des jetons. Les jetons sont transmis à l'activité qui suit le noeud de fusion. Quand un jeton arrive dans cette activité, elle est activée, c'est à dire prête à être exécutée. Il est ainsi possible que cette activité soit activée une fois pour chaque chemin entrant dans le noeud de fusion.

Le motif *discriminateur* est similaire au précédent mais la première branche dont la dernière activité passe de l'état "active" à l'état "terminée" déclenche déjà l'activité qui suit le discriminateur. La terminaison des autres branches est ignorée, c'est à dire qu'elle ne provoque pas l'émission d'un jeton vers l'activité qui suit le motif.

Le motif *jonction de sélection de n parmi m* décrit un comportement qui se situe entre la fusion de synchronisation et le discriminateur. Ce motif permet de déterminer le nombre de jetons entrant nécessaires pour l'émission d'un jeton de sortie. Les jetons entrants sur-numéraires sont ignorés.

2.6.3 Les motifs structurels

Les motifs structurels décrivent des comportements qui sont habituellement absents des SGWs car considérés comme atypiques et peu structurés, contrairement à leur appellation. Il convient de les considérer plus comme des anti-motifs à éviter que comme des motifs à réutiliser dans un modèle.

Le motif *cycle arbitraire* permet la répétition d'un groupe d'activités dans un workflow. Cette boucle est qualifiée d'arbitraire car le segment de workflow concerné par la boucle peut avoir plusieurs points d'entrée ou de sortie. Ce motif est surtout utile pour visualiser des boucles complexes dans un seul diagramme ou sur un seul niveau de workflow ce que ne permet pas le motif de boucle plus traditionnel basé sur la répétition d'un bloc (workflow de niveau inférieur).

Le motif *terminaison implicite* correspond à la terminaison d'une branche de contrôle particulière par manque d'activités à exécuter. La présence de branches de contrôles qui se terminent en cul de sac est en général considéré comme une mauvaise pratique de modélisation.

2.6.4 Les motifs d'instances multiples

Les motifs d'instances multiples correspondent aux situations qui se produisent quand une même activité est instanciée plusieurs fois et que ces instances s'exécutent en parallèle avec un besoin de synchronisation à la fin de leur exécution. Le nombre d'instances peut être soit déterminé à la modélisation soit à l'exécution. S'il est déterminé à l'exécution, il peut être soit fixé avant que les instances ne soient créées soit changer dynamiquement au fur et à mesure des instanciations. Ces différents cas de figure amènent différents motifs :

- instances multiples sans synchronisation (motif N°12)
- instances multiples avec connaissance à priori de leur nombre à la modélisation (motif N°13)
- instances multiples avec connaissance à priori de leur nombre à l'exécution (motif N°14)
- instances multiples sans connaissance à priori de leur nombre à l'exécution (motif N°15)

2.6.5 Les motifs basés sur des états

Les motifs basés sur des états dépeignent des situations dans lesquelles l'état global du workflow entre en jeu. Cet état peut être modifié par des événements externes au SGW.

Le motif *choix différé* (deferred choice) est proche du motif de choix exclusif en ce sens qu'il consiste en la sélection d'une seule branche parmi celles issues d'un noeud de décision. Dans ce motif, la sélection ne se base pas sur la valeur de vérité d'une garde mais sur la survenue d'un événement issus de l'environnement. Cet événement est par exemple généré pas un utilisateur du SGW qui sélectionne explicitement le flux de contrôle à suivre dans le workflow. Tous les autres chemins alternatifs sont exclus par l'événement.

Le motif *routage parallèle imbriqué* (Interleaved Parallel Routing) décrit une situation dans laquelle l'ordre des activités concernées n'a pas d'importance mais ou elles ne peuvent pas se dérouler en parallèle parce que par exemple elles utilisent toutes la même ressource non-partageable. Ce sont les agents en charge de ces activités qui choisissent l'ordre dans lequel ils vont les effectuer.

Le motif *étape* (Milestone) décrit un mécanisme qui permet d'influencer l'exécution du workflow à partir de l'occurrence de certains événements qui correspondent à des étapes du processus modélisé. Concrètement, la terminaison de certaines activités est requises pour que d'autres activités distantes (non directement liées par des transitions) puissent être activées.

2.6.6 Les motifs d'annulation

Les motifs d'annulation correspondent à l'annulation d'une activité ou d'une instance d'exécution du workflow.

Le motif *annulation d'une activité* décrit la situation dans laquelle une activité est annulée par l'occurrence d'une autre activité ou d'un événement externe au workflow.

Le motif *annulation d'un cas* décrit la situation dans laquelle c'est une instance complète d'un workflow qui est annulée.

2.6.7 Conclusion

Les diagrammes d'activités offrent une représentation intuitive des motifs de base. Pour des motifs plus "exotiques", ces mêmes diagrammes d'activités utilisent des symboles supplémentaires qui font appel à une sémantique opérationnelle subtile. Ils sont ainsi beaucoup moins intuitifs. La plupart des langages de workflow ont d'ailleurs de la peine à exprimer simplement ces motifs. C'est la raison pour laquelle l'équipe de van der Aalst a imaginé YAWL (Yet Another Workflow Langage), un langage conçu spécifiquement pour exprimer facilement ces motifs (voir section 3.3). Avec l'arrivée de la deuxième vague de motifs, le langage YAWL dans sa version initiale s'est révélé incapable de les exprimer tous et a dû être lui-même étendu [77].

Selon l'expérience de l'auteur, les motifs simples suffisent pour représenter la grande majorité des procédures que l'on rencontre dans le milieu hospitalier. Une syntaxe telle que celle des diagrammes d'activités avec éventuellement quelques extensions, notamment pour expliciter des contraites temporelles se révèle suffisante. Elle a en outre le mérite de rester intuitive pour le profane, contrairement à des langages plus pointus comme YAWL. Cette dernière caractéristique est très importante lorsque l'on doit "discuter workflows" avec des soignants qui ont au départ peu d'intuition dans ce domaine.

2.7 Résumé

Ce chapitre a introduit le lecteur aux principaux concepts issus du domaine des workflows et des systèmes de gestion de workflow. Il a également esquissé les étapes du cycle de vie d'un workflow depuis l'analyse préliminaire jusqu'à son incorporation dans un cycle d'amélioration continue. Il a insisté sur le rôle important joué par la Workflow Management Coalition dans la proposition d'un modèle de référence. Enfin, le concept de motif de workflow a été présenté et les 20 motifs initiaux proposés par van der Aalst et son équipe ont été brièvement expliqués.

Les sources suivantes ont été consultées pour l'écriture de ce chapitre :

- la description du modèle de référence de la Workflow management Coalition ainsi que le glossaire qui l'accompagne [34, 99].
- Les deux articles fondateurs de van der Aalst consacrés aux motifs de workflow[88, 90].
- L'article de White pour la description de la sémantique de certains motifs à l'aide de jetons et leur représentation sous forme de diagrammes d'activités UML [100].
- Le site internet consacré à l'outil ProM pour les explications relatives aux possibilités du Process Mining : http ://prom.win.tue.nl/research/wiki
- la figure qui esquisse le cycle de vie d'un workflow est adaptée d'une figure située à la page 86 de l'ouvrage de M. Muehlen [52].

Si la Workflow management Coalition propose un modèle de référence et définit les principaux concepts du domaine, elle ne dit en revanche rien sur la nature du contenu des modèles de workflows. Le chapitre suivant explore les différents formalismes utilisables pour représenter des workflows.

Chapitre 3

Représentation formelle des workflows

Malgré l'aspect familier et la nature prosaïque des workflows, le développement de modèles formels destinés à les représenter se révèle être un défi de recherche qui n'est pas encore totalement relevé. Le modèle de référence de la WfMC (voir section 2.5) définit un vocabulaire et identifie les interfaces d'entrée-sortie d'un système de gestion de workflow. En revanche, il ne dit rien de la nature d'un modèle formel de workflow. C'est la raison pour laquelle, malgré ces efforts de standardisation, plusieurs familles de langages formels basés sur des fondations théoriques différentes sont utilisés pour représenter des workflows. La liste suivante n'est sans doute pas exhaustive :

- les langages basés sur des graphes : réseaux de Petri, machines à état, diagrammes d'activité, ...
- les langages basés sur la logique : règles événement-condition-action, logique temporelle, logique transactionnelle, ...
- les algèbres de processus : pi-calcul, Orc, ...

Ces paradigmes théoriques sous-tendent, ou du moins inspirent, les principales normes et implémentations majeures dans ce domaine (tables 3.1, 3.2 et 3.3).

Les langages basés sur des graphes et sur des règles logiques sont les deux approches actuellement dominantes [47].

Les workflows sont naturellement représentés par des graphes dirigés dont les noeuds représentent les activités et les arcs les flux de contrôle ou de données entre les activités. Les modèles graphiques offrent une transcription visuelle du graphe et donc de la succession des activités qui composent le workflow. La plupart des langages de représentation de workflow basés sur la théorie des graphes sont liés de près ou de loin aux réseaux de Petri ou à certaines de leurs variantes.

La représentation d'un workflow sous la forme de règles est également assez naturelle. Il est vrai que les organisations édictent un nombre très important de règles pour assurer le bon fonctionnement de leurs processus internes.

Dans une approche de représentation de workflow à l'aide de règles, le flux de contrôle est abstrait sous la forme d'un ensemble de règles logiques. Une règle peut représenter une ou plusieurs activités du workflow. Dans ce paradigme, le système d'exécution du workflow est typiquement un moteur d'inférence qui détermine l'ordre d'exécution des activités du workflow en fonction de l'activation des règles. L'activation des règles survient en rép onse à des événements externes (dans l'environnement) ou internes au système de gestion de workflow.

Norme ou implémentation	Description	Organisme responsable
BPMN (Business Process Modelling Notation)	Représentation diagrammatique du workflow d'un processus métier. Assez similaire aux diagrammes d'activité UML.	OMG (Object Management Group)
BPEL (Business process Execution Language)	Langage exécutable de représentation de workflow utilisé principalement pour l'orchestration des services web (BPEL4WS). Une version adaptée à la représentation des processus métiers comportant des agents humains existe également (BPEL4People)	OASIS
YAWL (Yet Another Workflow Language)	Langage d'origine académique et son implémentation conçus pour exprimer et exécuter les principaux motifs de workflows (voir section 2.6)	Business Process Management (BPM) Center
UML 2.0 Activity Diagrams	Représentation diagrammatique de workflow utilisée pour représenter des processus métiers, des cas d'utilisation, des scénarios d'utilisation ou la logique d'une règle métier.	OMG (Object Management Group)

TAB. 3.1 – Principales normes et implémentations basées sur les réseaux de Petri

Implémentation	Description	Société responsable
ILOG BPM	Système de gestion de workflow amélioré grâce aux fonctionnalités d'un moteur de règles basé sur le paradigme E-C-A	IBM
Drools Business Logic Integration Platform	Système de gestion de workflow (Drools Flow) amélioré grâce aux fonctionnalités d'un moteur de règles (Drools expert) et d'un moteur de traitement d'événements et de raisonnement temporel (Drools Fusion)	JBoss
iProcess Suite	Système de gestion de workflow orienté-objectifs (Conductor) amélioré grâce aux fonctionnalités d'un moteur de règles (Decisions)	Tibco

TAB. 3.2 – Principales implémentations basées sur des règles logiques

3.1 Intérêt des langages formels

Par langage formel, on entend un langage dont la syntaxe et la sémantique sont définies de façon non-ambiguë, de préférence en se basant sur des concepts mathématiques (par exemple la théorie des graphes ou la logique formelle). Outre le fait d'éviter l'ambiguïté, l'utilisation de tels langages ouvre la porte à l'analyse et à la vérification des modèles décrits. Ils garantissent également une indépendance par rapport à la technologie qui les implémente.

Dans le cas de workflows, l'analyse permet d'anticiper le comportement d'un modèle avant son déploiement sur le terrain, par exemple en terme de performance, de consommation de ressources ou de charge de travail pour les agents. Dans certains cas, l'analyse consiste à exécuter le modèle dans un mode de simulation, c'est à dire découplé de la réalité du terrain.

La vérification permet d'apporter la preuve qu'un modèle de workflow présente certaines propriétés souhaitées. Ces propriétés peuvent couvrir différents aspects du workflow : confidentialité, sécurité, robustesse. Les propriétés les plus sujettes à vérification sont, en général, liées aux aspects de sécurité (safety) du workflow. Ce point est particulièrement important dans le monde médical où l'on souhaite s'assurer qu'un processus de soins ne mettra à aucun moment la vie d'un patient en danger ou ne lui portera préjudice.

3.2 Les réseaux de Petri

Les réseaux de Petri classiques et leurs dérivés sont sans doute les formalismes les plus utilisés pour représenter des workflows.

Un réseau de Petri est un modèle graphique dont la sémantique est basée sur des concepts mathématiques qui le rendent facilement analysable. Ce formalisme est utilisé pour représenter les systèmes

Norme	Description	Organisme responsable
WS-CDL (Web Services Choreography Description Language)	Langage d'orchestration de services web qui spécifie les collaborations pair à pair entre processus participants. ce langage permet de spécifier les comportements globaux et complémentaires et l'ordre des messages échangés dans la réalisation d'un objectif commun.	W3C
WSCI (Web Service Choreography)	Langage de description de processus métier pour l'orchestration de web services	W3C
XLANG	Langage exécutable de représentation de workflow utilisé principalement pour l'orchestration des services web, fortement similaire et concurrent de BPEL.	Microsoft
BPML (Business Process Modeling Language)	Meta-langage de description de processus métier abandonné par le BPMI en faveur de BPEL4SW (BPEL pour les services web)	OMG (Object Management Group)
BPEL	Langage exécutable de représentation de workflow utilisé principalement pour l'orchestration des services web (BPEL4WS). Une version adaptée à la représentation des processus métiers comportant des agents humains existe également (BPEL4People)	OASIS

TAB. 3.3 – Principales normes basées sur le pi-calcul

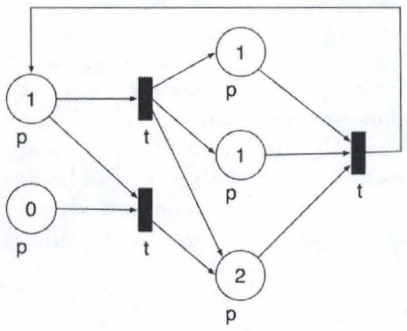

FIG. 3.1 – Un exemple de réseau de Petri

dynamiques en général. Les systèmes de gestion de workflows ne sont qu'un cas particulier de ce type de système.

Sous leur forme "classique", les réseaux de Petri existent depuis les années soixante. Ils furent imaginés par C.A. Petri dans sa thèse de doctorat intitulée "Kommunication mit Automaten" [67]. Grâce à leur généralité et leur aptitude à représenter la concurrence, les réseaux de Petri ont depuis été utilisés dans des domaines très divers : la modélisation, l'analyse et l'évaluation de performance de protocoles de communications, de systèmes distribués, de schémas d'orchestration de services web, ...

Les réseaux de Petri peuvent être considérés comme une variante des machines à état, également utilisable pour représenter des workflows (voir section 3.4.1).

Depuis leur invention, de nombreuses extensions leur ont été apportées pour renforcer leur expressivité et faciliter la modélisation.

3.2.1 Les réseaux de Petri classiques

Un réseaux de Petri est composé de *places* et de *transitions*. Les places peuvent contenir un certain nombre de *jetons*. Ce nombre peut-être nul. Les transitions sont des arcs dirigés qui connectent les places les unes aux autres. Chaque transition relie un ensemble de places d'entrée à un ensemble de places de sortie.

On associe à un réseau de Petri un marquage initial qui indique le nombre de jetons présents dans chaque place. Lorsqu'on exécute le modèle, le marquage évolue de façon discrète par un mécanisme de déclenchement des transitions. Lorsqu'une transition se déclenche, elle consomme un jeton dans chacune de ses places d'entrée et produit un jeton dans chacune de ses places de sortie. Une transition ne peut se déclencher que quand elle est est activée, c'est à dire quand chacune de ses places d'entrée contient au moins un jeton. Quand plusieurs transitions sont activées en même temps, n'importe laquelle d'entre-elles peut se déclencher de façon non déterministe. Les transitions ne peuvent pas se déclencher en même temps.

On représente graphiquement un réseaux de Petri en utilisant des boîtes pour les transitions et des cercles pour les places. Les relations de flux de contrôle dans le réseau sont représentées par des flèches qui connectent les places et les transitions. Le marquage est représenté par des points ou des chiffres qui symbolisent les jetons contenus dans les places.

Cette représentation graphique est facultative et on peut donner aux réseaux de Petri une définition purement formelle et mathématique :

> Un réseau de Petri R est un tuple $R = (P, T, F, M_0)$ où P est un ensemble de places, T est un ensemble de transitions, $F \subseteq (P \times T) \cup (T \times P)$ est la relation de flux de contrôle entre les places et les transitions. M_0 es une fonction de P vers $\mathbb{N}^+$ qui à chaque place associe un nombre entier positif et qui décrit le marquage initial du réseau. On représente l'évolution d'un réseau par une séquence de marquages $(M_0, M_1, \ldots, M_n)$ qui correspondent aux états successifs du réseau. Une paire de marquages successifs (M_i, M_j) représente le déclenchement d'une transition. Cette paire est valide si et seulement si il existe une transition activée en M_i et que le marquage des places d'entrée de cette transition est décrémenté de un entre M_i et M_j et que le marquage des places de sortie de cette transition est incrémenté de un entre M_i et M_j.
> (définition adaptée de la page 5 du rapport de Oren et Haller [61])

Comme un réseau de Petri est un modèle abstrait, on peut lui donner différentes interprétations en fonction du domaine d'application visé. Ainsi, les concepts de places et de transitions peuvent avoir différentes significations.

Dans la modélisation de systèmes réactifs, les places d'entrée et de sortie représentent les pré- et les post-conditions de survenue d'événements. Le déclenchement d'une transition représente la survenue d'un événement. Dans la modélisation de processus, les places d'entrée et de sortie peuvent représenter des ressources qui sont consommées et produites par une activité. Les transitions représentent la réalisation des activités. Dans la modélisation de systèmes dataflow, les places d'entrée et de sortie représentent des données d'entrée et de sortie. Les transitions représentent des étapes de calcul. L'ensemble des interprétations possibles d'un réseau de Petri est loin d'avoir été épuisé.

Indépendamment de ces différentes interprétations, la sémantique du réseau de Petri sous-jacent reste toujours la même ce qui garantit son comportement. Cette garantie de comportement est une propriété très importante des systèmes formels. A elle seule elle justifie l'effort consacré à leur mise en oeuvre.

3.2.2 Analyse des réseaux de Petri

Les réseaux de Petri supportent des techniques d'analyse qualitative et quantitative variées.

Les techniques qualitatives permettent la détection d'éventuels blocages (deadlocks), de branches mortes (non atteignables) ou d'accumulation de jetons à certaines places.

L'analyse des propriétés quantitatives liées à des facteurs de performance, de temps d'exécution ou de consommation de ressources font en général appel à de la simulation.

A cet effet, il existe des outils tels :

- Design/CPN de l'université d'Aarhus [26]
- ExSpect [29]
- ProM [70]

Néanmoins, ces résultats d'analyse peuvent-être considérés d'assez bas niveau eu égard à la problématique de la modélisation des workflows.

3.2.3 Les réseaux de Petri étendus

Comme signalé, de nombreuses extensions ont été apportées aux réseaux de Petri pour faciliter la modélisation ou améliorer leur expressivité. Sans prétendre à l'exhaustivité, en voici quelques-unes

parmi les principales : les réseaux de Petri colorés, les réseaux de Petri pondérés, les réseaux de Petri hiérarchiques et les réseaux de Petri temporisés. Ces extensions sont très utiles lorsqu'il s'agit de modéliser des workflows à l'aide de réseaux de Petri.

Les réseaux de Petri colorés associent des valeurs aux jetons, parfois sous la forme de structures de données élaborées. Ces valeurs permettent de représenter plus précisément les objets modélisés par les jetons et de considérer différents types d'objets. Les transitions ont le pouvoir de changer les valeurs associées aux jetons et donc d'agir sur les objets qu'ils représentent. Si un réseau de Petri coloré est utilisé pour représenter un workflow, par exemple dans le monde médical, les jetons peuvent représenter des patients qui transitent dans le workflow. Dans ce cas, les valeurs associées aux jetons correspondront à certaines caractéristiques des patients tel leur âge, leur tension artérielle ou leur taux de cholestérol. Les activités du workflow agissent potentiellement sur les caractéristiques du patient. Par exemple, une activité thérapeutique peut faire baisser la tension artérielle du patient.

Les réseaux de Petri pondérés associent un poids aux arcs qui représentent les transitions. Ces poids indiquent le nombre de jetons consommés ou produits lors du déclenchement de la transition. Si un réseau de Petri est utilisé pour modéliser un workflow et que les jetons représentent une ressource produite ou consommée, la possibilité d'ajouter des poids sur les arcs facilite cette modélisation. En effet, un arc avec un poids p dans un réseau de Petri pondéré correspond à p arcs parallèles dans un réseau de Petri classique. La notation pondérée est donc beaucoup plus compacte.

Les réseaux de Petri hiérarchiques introduisent la notion de sous-réseau qui permet une modélisation modulaire qui facilite la compréhension, la maintenance et la réutilisation des modèles.

Enfin, les réseaux de Petri temporisés permettent de modéliser les comportements influencés par le passage du temps. Dans ces modèles, on associe à chaque jeton une balise temporelle qui indique à quel moment le jeton peut-être consommé. On associe également un délai à chaque transition. Ce délai sert à incrémenter la valeur des balises temporelles des jetons qui transitent par ces transitions. La représentation d'aspects liés à l'écoulement du temps est très important dans les modèles de workflows. Ceci est particulièrement vrai pour les workflows médicaux dans lesquels les contraintes temporelles imposées sont en général liées à des aspects critiques pour le patient. La prise en compte d'aspects temporels permet également d'effectuer des analyses ou des simulations intéressantes quant à la performance du workflow : temps de réponse, occurrence de dépassement de délais, temps minimaux et maximaux pour exécuter les activités ou l'ensemble du workflow, ...

3.2.4 Les réseaux de workflow

Il existe une classe particulière de réseaux de Petri appelés réseaux de workflow (workflow nets) [87] qui, comme leur nom l'indique, sont bien adaptés à la représentation de workflows.

Les réseaux de workflows ont deux propriétés particulières qui les distinguent des autres réseaux de Petri :

- Ils ont exactement une seule place d'entrée et une seule place de sortie.
- Chaque transition et chaque place sont situées sur un chemin qui relie la place d'entrée à la place de sortie du réseau.

3.2.5 Modélisation de workflows à l'aide de réseaux de Petri

Les qualités des réseaux de Petri comme langage de modélisation de workflow tiennent à leur nature graphique, à leur sémantique formelle et à l'abondance de techniques qui permettent de les analyser [86].

Plusieurs stratégies sont envisageables pour représenter un workflow à l'aide d'un réseau de Petri [61]. Ces stratégies influencent la façon dont sont représentés les événements qui font progresser le workflow et les activités qui le constituent.

Un système de gestion de workflow doit réagir à des événements du monde extérieur pour faire évoluer l'état de ses modèles internes. Il existe au moins deux façons de représenter des événements dans le formalisme des réseaux de Petri : par des jetons ou par des transitions.

On peut représenter les événements sous la forme de jetons. Dans ce cas, on réserve une place pour tout événement potentiel. Un jeton y sera placé à l'occasion de l'événement en question dans l'environnement. Le problème avec cette représentation est qu'il devient difficile de propager les événements puisque comme jetons ils sont directement consommés par les transitions. On peut également modéliser des événements sous forme de transition. Une transition est ainsi créée dans le réseau de Petri pour tout événement potentiel.

Nous avons vu que les activités constituent les briques de base d'un modèle de workflow. Tout comme les événements, il existe au moins deux façons de représenter des activités dans un réseau de Petri : par des transitions ou par des places.

On peut représenter les activités à l'aide de transitions. On crée pour chaque activité du workflow une transition dans le réseau de Petri. L'exécution d'une activité est représentée par le déclenchement de la transition correspondante. Cette façon de faire soulève certains problèmes :

- Dans la sémantique des réseaux de Petri, les transitions sont instantanées alors que dans le contexte d'un workflow, les activités ont en général une certaine durée. Une solution consiste à prévoir une transition pour le début de l'activité et une transition pour la fin.
- La plupart des activités du workflow ne sont pas exécutées par le SGW mais bien par les agents qui font partie de l'environnement. Certaines activités dites "automatiques" sont néanmoins exécutée par le SGW. En représentant les activités par des transitions il est difficile de faire la distinction entre des transitions qui représentent des activités effectuées dans l'environnement et des transitions qui représentent des activités automatiques.
- Dans certain cas, un SGW doit se comporter comme un système réactif, par exemple en déclenchant une alarme quand une condition particulière se présente. Or, selon la sémantique des réseaux de Petri, une transition activée peut se déclencher sans nécessairement avoir l'obligation de le faire. Ce comportement va à l'encontre du comportement classique d'un système réactif.

On peut également représenter les activités par des places. Dans cette représentation, il existe une place par activité du workflow. La présence d'un jeton dans une place indique que l'activité correspondante est en cours. Une limitation liée à cette représentation est que l'exécution des activités peut modifier l'état de l'instance courante du workflow (représentée par des jetons éventuellement colorés) ce que les places ne permettent pas de faire selon la sémantique des réseaux de Petri.

Le lecteur intéressé par les techniques avancées de représentation de workflows à l'aide de réseaux de Petri consultera l'article suivant de van der Aalst et son équipe [90]. Cette même équipe relève trois problèmes principaux dans la représentation de workflows à l'aide de réseaux de Petri étendus. Elle relie ses problèmes à l'absence de support de trois classes particulières de motifs par les réseaux de Petri : les motifs d'instances multiples, les motifs de branchements avancés et les motifs d'annulation [89].

3.2.6 Conclusion

En guise de conclusion, il est possible et même souhaitable de représenter un workflow à l'aide d'un réseau de Petri eu égard à leur formalisme bien fondé et aux techniques d'analyse disponibles. Néan-

moins, un réseau de Petri est un formalisme d'assez bas niveau qui nécessite de la part du modélisateur un effort non négligeable dés qu'il s'agit de représenter des comportements complexes ou des motifs avancés. Les extensions apportées aux réseaux de Petri (couleur, poids, temps, hiérarchie) apportent certaines facilités mais ne sont pas suffisantes pour exprimer tous les motifs. C'est la raison pour laquelle un nouveau langage, YAML a été récemment introduit.

3.3 YAWL

L'appellation du langage YAWL, "Yet Another Workflow Language", illustre bien la pléthore de représentations plus ou moins formalisées qui existent dans ce domaine. Ce langage à été conçu par des laboratoires issus de la Eindhoven University of Technology et de la Queensland University of Technology [89]. Il s'agit des mêmes équipes qui sont à l'origine des premières publications sur les motifs de workflow.

YAWL est né précisément du besoin d'exprimer de façon concise et intuitive un maximum de motifs de workflows. Il pallie ainsi aux faiblesses d'autres langages utilisés dans ce domaine, notamment les réseaux de Petri.

YAWL est construit initialement à partir des réseaux de Petri et plus spécifiquement sur les réseaux de workflow (voir section 3.2.4) auxquels il rajoute le support :

- d'instances multiples
- d'activités composites
- de jonctions-OU
- de la suppression de jetons
- des transitions directement connectées sans interposition de décisions

La sémantique de YAWL est indépendante de celle des réseaux de Petri. Pour le lecteur intéressé, elle est décrite sous la forme d'un système de transitions dans l'article publié par van der Aalst en 2005 [93].

Un modèle de workflow exprimé en YAWL est un réseau composé d'activités et de décisions. Ce réseau peut-être éventuellement hiérarchique, c'est à dire composé de sous-réseaux. Les activités sont soit atomiques et ne plus être décomposables en sous-activités, soit composites et faire référence à un autre sous-réseau.

Le réseau est normalement constitué d'une alternance d'activités (comparables aux transitions des réseaux de Petri) et de décisions (comparables aux places des réseaux de Petri). Il est possible que des activités soient connectées directement entre elles, sans interposition de décisions.

Les motifs de base de flux de contrôle sont supportés par défaut dans le langage. Il existe ainsi des activités qui représentent la division et la synchronisation parallèle, le choix et la synchronisation exclusive.

Comme signalé à la section 3.2.5, il existe trois principaux problèmes quant à la représentation de workflows à l'aide de réseaux de Petri. Ces problèmes sont liés à l'absence de support de trois sous-groupes de motifs. YAWL lève ces trois problèmes en offrant un support direct de ces trois sous-groupes dans le langage.

Les motifs relatifs aux instances multiples d'activités sont directement supportés. La notation YAWL permet de spécifier le nombre minimum et maximum d'instances permises par activités et si ce nombre peut-être déterminé à l'exécution ou doit être connu à la création du modèle.

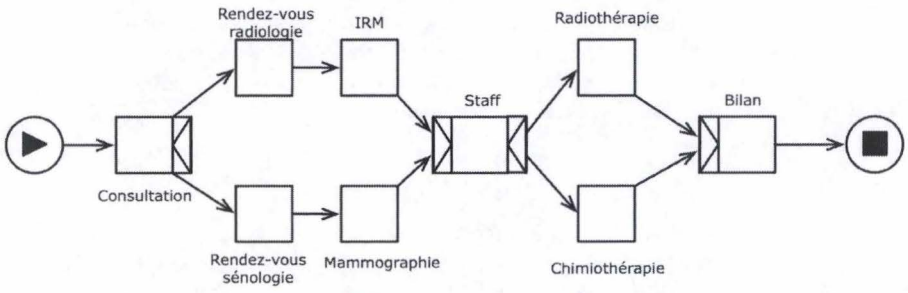

FIG. 3.2 – Aspect visuel d'un workflow simple en YAWL

Les motifs d'annulation sont également supportés par la notation. YAWL permet d'indiquer la suppression de jetons. On peut connecter une activité à un certain nombre de noeuds de condition et indiquer que lorsque cette activité s'exécute, les jetons contenus dans ces noeuds sont retirés.

Enfin, les motifs de branchement avancés sont supportés par la mise à disposition de noeuds de division et de jonction-OU non exclusives.

Au total, YAWL supporte l'expression de 19 des 20 motifs de flux de contrôle. Le motif n°11 (terminaison implicite) n'est pas supporté mais comme signalé à la section 2.6.3, il doit être considéré comme un anti-motif. Son usage n'est donc pas recommandé.

3.4 Autres approches basées sur les graphes

3.4.1 Les machines à états

David Harel est l'auteur des machines à états modernes (statecharts) [31] qui permettent aux actions de se dérouler à la fois au niveau des états et au niveau des transitions et supportent les notions d'états hiérarchiques et concurrents. Les diagrammes d'états d'UML sont fortement influencés par les statecharts de Harel.

Les machines à états, de part leur approche orientée-état, se distinguent assez fondamentalement des représentations qui traitent un workflow comme une succession d'activités. La première étape dans la modélisation consiste à déterminer l'objet représenté par la machine à état (par exemple un patient, voir figure 3.3). La seconde étape consiste à énumérer les états par lesquels cet objet est susceptible de transiter (malade, diagnostiqué, traité, suivi, guéri). Ensuite vient la détermination des transitions possibles entre ces états, des événements susceptibles de provoquer ces transitions. Enfin, sont décidées les actions à effectuer tant au niveau des états que des transitions.

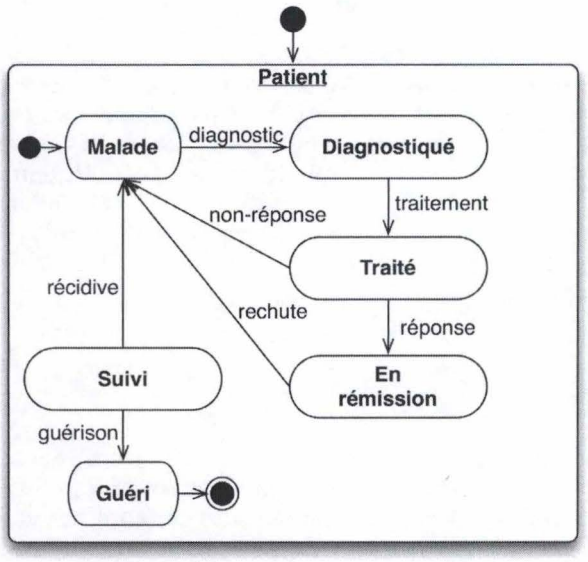

FIG. 3.3 – Un exemple de machine à état de type Statechart qui représente un patient

Telles quelles, les machines à états sont peu utilisées pour représenter directement des workflows. Par contre, certaines approches transforment des représentations de plus haut niveau tels des diagrammes d'activités, en machine à états pour bénéficier des outils d'analyse disponibles. Dans la publication suivante [50], les auteurs représentent directement un processus de soins en oncologie (traitement du cancer) sous forme d'automates temporisés (une classe particulières de machine à états). Ils utilisent ensuite un vérificateur de modèle (model checker), baptisé Uppaal [7], pour vérifier que leur modèle possède bien une série de propriétés liées au domaine. Ces propriétés sont notamment en relation avec des contraintes temporelles fortes imposées par le bon déroulement du traitement. Cette approche est intéressante mais présente une limitation importante : les workflows sont représentés dans un formalisme de très bas niveau qui, s'il est directement utilisable par les outils de vérification, demeure hors de compréhension des experts du domaine (les soignants, dans ce cas). L'article suivant [21], auquel a collaboré l'auteur de ce texte, utilise une représentation de haut niveau sous forme de hMSC (un formalisme proche des diagrammes d'activités d'UML) et de MSC [35] (proche des diagrammes de séquences d'UML) pour représenter des workflows médicaux. Cette représentation est transformée en une représentation de bas niveau de type LTS ("Labelled Transition System", une forme particulière de machine à états) pour faire ensuite l'objet d'analyses poussées (exécution symbolique, model checking).

3.4.2 Les diagrammes d'activités

Les diagrammes d'activités UML sont proches de la vision diagrammatique intuitive de type "organigramme" que l'on retrouve le plus souvent pour représenter des processus avec un faible niveau de formalisme. Ce style de représentation est très répandu, sans doute parce qu'il est proche de celui de la programmation procédurale. Néanmoins, l'utilisation des diagrammes d'activités comme modèle formel de workflow est peu répandu. Le langage graphique BPMN, assez similaire dans ses concepts,

leur est largement préféré.

A la différence des machines à états, les diagrammes d'activités traitent un processus comme une succession d'activités. Il existe un ambiguïté gênante dans leur définition sémantique. Dans les versions 1.x d'UML, les diagrammes d'activités sont définis comme une variation des diagrammes d'états UML. Selon cette définition, les états représentent des activités et les transitions représentent la terminaison de ces activités. Cette interprétation, qui dénature la signification des diagrammes d'états, a été vivement critiquée. Dans la version 2.0 de la spécification d'UML [60], les diagrammes d'activités ont une sémantique basée sur la transmission de jetons, proche de celle des réseaux de Petri (voir section 2.6).

3.4.3 Les arbres de tâches

La représentation d'un workflow sous la forme d'un arbre de tâches repose sur le principe suivant : les activités qui composent le workflow sont soit atomiques soit composites (composées de plusieurs sous-activités). Les activités peuvent donc être organisées de façon hiérarchique (voir la figure 3.4). Dans cette représentation, les workflows sont progressivement segmentés en leurs sous-activités qui constituent les feuilles d'un arbre. Les embranchements de l'arbre sont des structures de contrôles (exécution parallèle, séquentielle, décisions, ...).

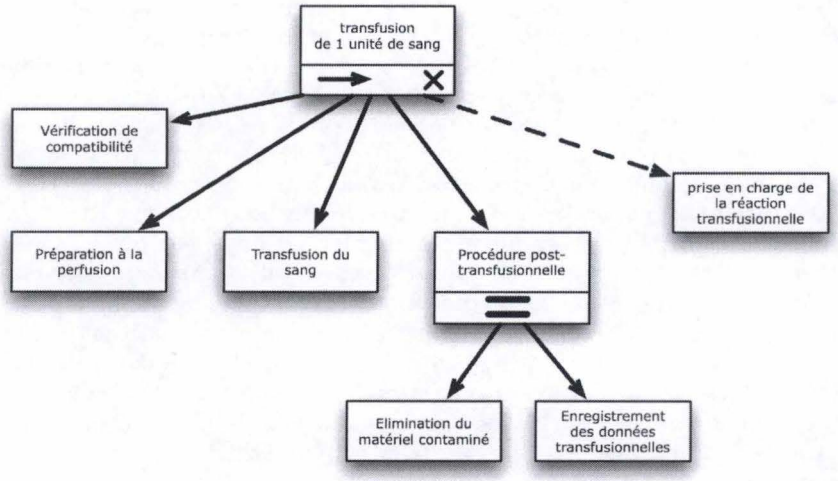

FIG. 3.4 – Exemple de représentation d'une procédure de transfusion sanguine sous la forme d'un arbre de tâches (adapté de Osterweil [62])

Little-JIL [15] est un langage de description de processus basé sur ce principe. Il a notamment été appliqué à la représentation de workflows médicaux dans les domaines de la transfusion sanguine et du traitement du cancer.

3.5 Représentation d'un workflow à l'aide de règles

L'idée de représenter des processus à l'aide de règles trouve son origine dans le monde des bases de données et plus particulièrement des bases de données actives. Une base de donnée active possède certains mécanismes qui lui permettent de répondre à des événements internes ou externes. Les réponses à ces événements sont encodées sous la forme de règles événement-condition-action (ECA). Le lecteur intéressé peut consulter l'article de Paton [64] qui offre une bonne vue d'ensemble de ce sujet.

On retrouve également des règles ECA dans des architectures orientées-événements tels les systèmes de traitement d'événements (event stream processing systems) utilisés dans le monde financier ou dans les réseaux de capteurs. Les moteurs de règles métiers (business rules engine) tels que les systèmes de production manipulent également ce type de règles.

Une règle ECA est composée de trois parties :

- une partie événementielle
- une partie conditionnelle
- une partie opérationnelle

Voici la représentation d'une règle ECA simple :

> **quand** un certain *événement* se produit
> **si** une certaine *condition* est vraie
> **alors** effectue une certaine *opération*

En toute généralité, il peut y avoir plusieurs parties conditionnelles et opérationnelles dans une même règle ce qui la rapproche des structures de contrôle conditionnelles des langages de programmation :

> **quand** un certain événement e se produit
> **si** une condition c_1 est vraie
> **alors** effectue l'opération o_1
> ...
> **sinon si** une condition c_n est vraie
> **alors** effectue l'opération o_n
> **sinon** effectue l'opération o_0

La dernière opération o_0 s'exécute si aucune des conditions précédentes n'est d'application. L'exécution d'une opération peut provoquer la survenue d'un événement. Ce mécanisme permet de lier les règles entre elles pour les exécuter en cascade (voire figure).

Les parties conditionnelles et opérationnelles sont facultatives : l'événement déclenche toujours l'opération (condition vide) ou l'événement n'a aucun effet (condition et opération vide).

La partie événementielle peut contenir plus que la référence à un événement simple. Des motifs d'événements complexes sont également possibles :

la sélection d'événements : m événements parmi la liste $\{e_1, e_2, \ldots, e_n\}$ avec $m \leq n$

la séquence d'événements : les événements $\{e_1, e_2, \ldots, e_n\}$ selon un séquencement particulier

des événements périodiques : toutes les n survenues de l'événement e

des intervalles de temps : l'événement e dans l'intervalle de temps compris entre deux autres événements e_1 et e_2

Prises dans cette formulation générale, les règles ECA permettent de représenter sans difficulté les motifs de contrôle de base des workflows.

Pour représenter une séquence d'activités (fig. 3.5), il suffit de lier les règles entre elles à l'aide de leurs opérations et de leurs événements : l'exécution de l'opération d'une règle provoque l'événement déclencheur de la suivante et ainsi de suite. Dans la figure suivante, les flèches matérialisent le flux de contrôle qui résulte de l'enchaînement des règles.

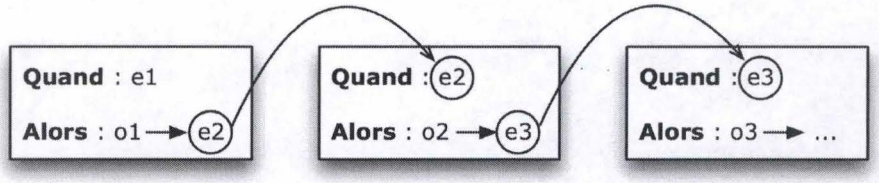

FIG. 3.5 – Représentation du motif de séquence sous forme de règles ECA liées entre elles par leurs opérations et leurs événements (adapté de [40])

L'implémentation des autres motifs repose sur le même principe. Un motif de division parallèle (division-ET) peut-être représenté :

- soit en provoquant en une seule opération tous les événements déclencheurs des branches de la division parallèle.
- soit en ne provoquant qu'un unique événement déclencheur de toutes les branches.

Un motif de synchronisation parallèle peut-être représenté lorsque chaque branche provoque un événement et que la conjonction de ceux-ci déclenche une règle unique.

Le choix exclusif fait appel à une règle dont la partie conditionnelle contient plusieurs conditions. La synchronisation exclusive utilise une règle dont la partie événementielle est une disjonction d'événements. Enfin, l'itération fait appel à l'émission d'un événement récurrent pour ré-exécuter plusieurs fois le corps de la boucle. La condition de sortie est exprimée dans la partie conditionnelle d'une règle.

Des événements peuvent être provoqués par des temporisateurs. Ce mécanisme permet de spécifier des contraintes de temps absolues (liées à un événement provoqué à un instant t bien défini), des contraintes de temps relatives (liées à un événement provoqué un certain temps après un autre événement) ou des répétitions temporelles (liées à un événement répété régulièrement sur un intervalle de temps $t_1 \leftrightarrow t_2$.

Enfin, certaines extensions apportées au contenu des règles ECA permettent de représenter les acteurs en charge de certaines opérations ou de certaines décisions ainsi que les données consommées par ces mêmes opérations et décisions. Le lecteur intéressé se référera à l'article de Knolmayer [40] pour plus de détails.

3.6 Les systèmes orientés-agents

Un agent est un système informatique situé dans un certain environnement et capable d'activités autonomes pour atteindre ces objectifs. Un agent rationnel ou intelligent doit en plus être réactif (répondre rapidement aux changements de son environnement), proactif (mener certaines activités pour atteindre ses objectifs) et social (interagir avec d'autres agents).

La proactivité est sans doute ce qui distingue le plus les systèmes orientés-agents des systèmes orientés-objets traditionnels. Bien que des objets puissent-être réactifs, ils ne sont pas proactifs dans le sens ou leurs réactions ne sont pas dictées par des objectifs explicites et persistants. Un objectif persistant continuera à influencer les activités de l'agent malgré plusieurs tentatives avortées pour l'atteindre.

Les activités d'un agent peuvent échouer dans leurs tentatives d'atteindre un objectif. Cet agent peut surmonter ces échecs s'il dispose d'un panel étendu d'activités ou de combinaisons d'activités pour atteindre un même objectif. Si une combinaison d'activités échoue, il peut en sélectionner une autre. Nous verrons que cette flexibilité dans la sélection des activités qui confère de la robustesse au systèmes orientés agents est également très importante pour les systèmes de gestion de workflow, particulièrement dans un contexte médical.

3.6.1 Activités et plans

Les activités ou actions sont les moyens par lequel un agent influence son environnement. Sous leur forme la plus simple, elles sont atomiques, peuvent réussir ou échouer et ont une certaine durée. Plus généralement, les activités peuvent être combinées de façon plus ou moins complexe sous la forme de plans prédéfinis. Les plans sont stockés dans une librairie de plans d'où ils peuvent être sélectionnés à l'aide de requêtes.

Un plan permet d'atteindre un certain objectif. Il est d'application dans un certain contexte (préconditions du plan). Il possède une définition opérationnelle qui correspond à la séquence coordonnée des activités qui le compose. Enfin, il contient parfois certaines informations supplémentaires qui permettent de raffiner les requêtes de sélection : coût occasionné par l'exécution du plan, probabilité de réussite, ressources consommées, ...

Les plans peuvent être considérés comme des fragments de workflows. Dans le monde des agents, ils sont d'ailleurs exprimés à l'aide de formalismes qui sont les même que ceux utilisés pour les workflows : depuis du code informatique de bas niveau à des formalismes basés sur des graphes ou des règles .

3.6.2 Événements et objectifs

Un agent représente un compromis entre réactivité (activités dictées en réaction à des changements dans l'environnement) proactivité (activités dictées par des objectifs internes). Deux abstractions permettent d'atteindre ce compromis : les événements et les objectifs.

Un événement est quelque chose qui survient et auquel un agent doit répondre de façon appropriée. Les événements peuvent survenir lors de changements dans l'environnement ou être générés par des composants de l'agent lui même, par exemple une horloge. Un événement peut déclencher une activité, modifier l'état interne de l'agent ou activer la prise en compte d'un nouvel objectif. Un agent qui ne tiendrait compte que des événements pour effectuer ses activités agirait de façon purement réflexe et serait incapable de comportements à plus long terme. C'est la raison pour laquelle une deuxième abstraction est nécessaire : l'objectif.

Un objectif est quelque chose que l'agent souhaite atteindre. Il s'agit souvent d'un certain état de son environnement. Plusieurs types d'objectifs sont envisageables :

- achieve : atteindre un certain état
- maintain : maintenir un certain état
- avoid : éviter un certain état (souvent utilisé pour exprimer des contraintes de sécurité)

Cette typologie d'objectif a également été retenue par certaines méthodes d'ingénierie des spécifications (voir section 6.3.1).

Ces sont les objectifs qui confèrent aux agents leur autonomie. Comme déjà signalé, ces objectifs sont persistants : si un plan échoue dans l'atteinte d'un objectif, un plan alternatif est sélectionné pour atteindre l'objectif en question.

Chaque objectif est potentiellement atteignable par un certain nombre de plans différents. A l'exécution, c'est l'agent qui choisira un plan pour atteindre un objectif donné. Si ce plan échoue, un autre plan sera essayé. De cette façon, les agents sont à la fois flexibles (il y a plus d'un plan possible pour atteindre un objectif) et robustes (l'échec d'un plan ne signifie pas nécessairement l'échec de l'objectif).

Ces caractéristiques de proactivité, réactivité, flexibilité et robustesse sont également souhaitables pour un système de gestion de workflows. Même si ce sont des agents externes au SGW, pour la plupart humains, qui exécutent les activités, c'est le SGW qui doit orchestrer leur bon déroulement et proactivement signaler aux agents les activités suivantes. Si une condition exceptionnelle se produit, le SGW doit agir réactivement pour avertir les agents et proposer éventuellement des activités de remédiations à cette exception. En cas d'échec d'un plan, le SGW ne doit pas laisser les agents livrés à eux-mêmes mais leur proposer un ou plusieurs plans alternatifs. Les informations associées aux plans permettent leur sélection raisonnée et d'offrir un service proche de celui d'un système d'aide à la décision.

C'est donc essentiellement la satisfaction d'objectifs qui est le principal moteur de l'exécution des systèmes orientés-agents. Ce concept peut-être également appliqué aux systèmes de gestion de workflows. C'est une des idées centrales de ce travail qui sera explorée au chapitre 6.

3.6.3 Les méthodes de conception de systèmes orientés-agents

Il existe une littérature abondante sur la conception de systèmes orientés-agents. Voici trois méthodologies parmi les plus utilisées [20] :

- Gaia [104]
- Prometheus [63]
- Tropos [11]

Ces trois méthodologies ont en commun une place importante faite à la modélisation des objectifs et des plans. Comme le rôle des objectifs est fondamental dans la description du comportement des agents, les méthodes de conception de systèmes orientés agents disposent en général de techniques avancées pour éliciter et représenter formellement des objectifs. En général, les objectifs y sont organisés selon une décomposition hiérarchique.

3.7 Autres approches basées sur la logique

3.7.1 Les logiques temporelles

Plusieurs approches sont possibles pour introduire une perspective temporelle dans la représentation logique d'un processus métier. La technique la plus immédiate consiste à étiqueter chaque variable du modèle avec une valeur de temps :

$Traitement(Dupont, Aspirine, t)$ ou t représente un certain instant.

Une technique plus évoluée consiste à adopter une logique dite temporelle. Ce genre de logique permet une description déclarative de l'évolution historique d'un système sans recourir à un étiquetage complexe à maintenir.

La logique temporelle moderne a été introduite par Prior à la fin des années 50 [69]. Elle a surtout été étendue au domaine de l'informatique par les travaux de Pnueli [68] pour la spécification et la vérification de systèmes réactifs et concurrents.

Il existe de nombreuses variantes de logiques temporelles qui diffèrent dans leur manière d'envisager l'historique du système considéré et le temps lui-même.

L'historique du système peut-être :

- une séquence linéaire potentiellement infinie d'états temporisés.
- une arborescence d'états dont les chemins comportent des embranchements alternatifs.

Deux variantes de logiques temporelles reposent sur cette distinction :

- la logique temporelle linéaire (Linear Temporal Logic, LTL)
- la logique temporelle ramifiée (Computation Tree Logic, CTL)

Le temps lui même peut-être représenté de différentes façons, soit en faisant référence à des instants (points temporels) soit en considérant des intervalles de temps (Interval Temporal Logic, ITL [14]). La structure même de la ligne du temps est sujette à variations. On peut considérer un temps :

- discret, constitué d'instants indivisibles et pour lesquels le passage d'un instant à l'autre est une transition irréductible. Ces instants discrets peuvent être projetés sur une série discrète de nombres naturels tels que chaque instant peut-être désigné par un entier positif.
- dense, dans lequel les instants peuvent-être projetés sur une série dense de nombre rationnels (série infinie dénombrable) avec pour corollaire qu'il existe toujours un instant entre deux instants considérés et qu'aucun instant n'est contigu à un autre.
- continu, où les instants sont projetés sur une série continue de nombres réels (infinie non-dénombrable).

Tant la LTL que la CTL sont des extensions de la logique du premier ordre. La LTL par exemple, rajoute des connecteurs temporels aux connecteurs logiques traditionnels [96] :

Passé	Futur
♦ : à un certain moment dans le passé	◊ : à un certain moment dans le futur
■ : toujours dans le passé	□ : Toujours dans le futur
B : toujours dans le passé jusqu'à ...	***W*** : Toujours dans le futur à moins que ...
S : toujours dans le passé depuis ...	***U*** : Toujours dans le futur jusqu'à ce que ...
• : dans l'état précédent	○ : dans l'état suivant

La LTL considère un système comme une séquence infinie d'états. Il existe une fonction "Historique" H qui projette un nombre naturel qui représente un instant sur chacun de ces états. La sémantique de la LTL repose sur cette notion d'historique. La satisfaction d'une formule LTL P est exprimée par rapport à l'historique H du système à un instant déterminé i :

$$(H, i) \models P$$

En CTL, l'historique du système est assimilé à une structure arborescente. Des embranchements alternatifs sont possibles pour des états successeurs. La notion de chemin se rajoute à celle d'état pour raisonner sur la validité d'une formule.

La logique temporelle, et plus particulièrement la CTL trouve sa place parmi les langages de représentation de workflows suite au travaux de Attie et de ses collègues. Ces recherches furent menées dans le domaine des base de données réparties et plus particulièrement à propos de la spécification des propriétés transactionnelles d'opérations coordonnées [61].

La logique temporelle est également utilisée pour exprimer des contraintes temporelles sur des workflows [28]. Les workflows sont en général représentés dans un formalisme de haut niveau (diagrammes d'activités UML) pour être ensuite transformés en un système transitionnel et être vérifié à l'aide d'un vérificateur de modèle (model checker).

Le framework KAOS [96] auquel ce travail fait référence, utilise des formules de la LTL pour spécifier formellement les exigences d'un système d'information en cours de conception.

3.7.2 La logique transactionnelle

La logique transactionnelle est une extension de la logique du premier ordre proposée par Bonner et Kifer au début des années nonantes [9]. On l'utilise pour spécifier et vérifier des séquences d'opérations coordonnées sur des bases de données dynamiques. Les concepts de transaction et d'opération atomique sont centraux dans cette logique et permettent de raisonner formellement sur l'évolution d'une base de donnée. Plusieurs publications sont consacrées à l'utilisation de la logique transactionnelle pour modéliser et vérifier des workflows [61].

3.7.3 Le calcul des événements

Le calcul des événements est décrit en détail à la section . Seulement deux travaux existent quant à son utilisation pour représenter des workflows :

- une approche procédurale basée sur la programmation logique [16, 17]
- une approche purement déclarative [55]

3.7.4 Le calcul situationnel

Le calcul situationnel est un formalisme de représentation des connaissances basé sur la logique du premier ordre. Tout comme le calcul événementiel, il permet de raisonner sur des scénarios dynamiques qui s'exécutent dans un monde changeant [74]. Il repose sur les concepts suivants :

- des actions
- des situations
- des objets
- des fluents

Les actions provoquent des changements d'état du monde représenté. Une situation est simplement un historique possible de ces changements d'états. Une situation est une séquence d'actions. Elle représente l'état du monde qui résulte de l'application de cette séquence d'actions. La constante S_0 représente la situation initiale du monde, soit une séquence d'actions vide.

Le calcul situationnel définit un symbole de fonction particulier : do. L'expression $do(\alpha, s)$ représente la situation qui suit la situation s après l'occurence de l'action α.

Les actions sont représentées par des symboles de fonctions qui peuvent comporter des paramètres :

> $traiter(p, m)$ représente l'action de traiter un certain patient p avec un médicament m, p et m étant des objets.
>
> $do(traiter(Dupont, Aspirine), s)$ représente la situation qui résulte du traitement de monsieur Dupont avec de l'aspirine depuis la situation s.
>
> $do(suivre(Patient), do(traiter(Cancer), do(opérer(Cancer), do(diagnostiquer(Patient), S0))))$: cette situation représente un scénario simple du traitement d'un cancer à partir de la situation initiale.

Pour représenter les changements qui s'opèrent dans le monde, le calcul situationnel fait appel au concept de fluent. Un fluent est une variable logique dont la valeur est susceptible de changer en fonction du temps.

Il existe deux types de fluents en calcul situationnel :

- les fluents relationnels représentent des relations dont la valeur de vérité change de situation en situation. Ils sont représentés par des symboles de prédicats dont le dernier argument est un terme qui représente une situation.
- les fluents fonctionnels représentent des fonctions dont la valeur change de situation en situation. Ils sont représentés par des symboles fonctionnels dont le dernier argument est un terme qui représente une situation.

> $tension(p, s)$ est un fluent fonctionnel qui représente la tension d'un certain patient p dans la situation s, c'est à dire l'état du monde qui résulte de la séquence d'actions de s.
>
> $contaminé(p, g, s)$ est un fluent relationnel qui exprime le fait qu'un patient p est contaminé par un germe g dans la situation s.

Le calcul situationel est assez proches du calcul événementiel [43]. Ce qui les distingue principalement est leur façon d'envisager les fluents. Dans le calcul situationnel, les fluents sont considérés comme des prédicats ou des fonctions dont un des arguments représente la suite des actions passées. Dans le calcul événementiel, les fluents sont représentés par des fonctions logiques du temps. Ainsi, la fonction $NeutrophilesBas(P)$ représente le fait que le patient P a un taux de neutrophiles (une sorte de globule blanc) bas ou pas en fonction du temps. Lorsqu'un fluent est représenté par une fonction, il faut utiliser un prédicat spécial pour spécifier quand le fluent est vrai ou pas. C'est l'approche choisie par le calcul événementiel. Ainsi, $HoldsAt(NeutrophilesBas(P), T)$ signifie que le patient P à un taux de neutrophiles bas au temps T.

Il existe une famille de langages de programmation logique basée sur le calcul situationnel : Golog, un langage de programmation logique orienté agents ; ConGolog, une version qui incorpore la programmation concurrente ; RGolog, une version réactive, ...

Le calcul situationnel mérite sa place parmi les langages formels de représentation de workflows, notamment grâce aux travaux de Koubarakis et ses modèles de processus métiers en Golog [41].

3.8 Les algèbres de processus

Bien que moins représentés, les algèbres de processus ont également été proposées pour modéliser formellement des workflows. Le terme "algèbre de processus" désigne une famille de langages utilisés pour modéliser des systèmes concurrents, éventuellement distribués. Ces langages reposent sur des lois algébriques (d'où leur appellation) qui permettent la manipulation des processus représentés. ces langages ont en commun de permettre une description formelle de haut niveau des interactions (communications, synchronisations) entre ces processus.

3.8.1 Le pi-calcul

Dans le domaine des workflows, parmi les algèbres de processus, c'est le pi-calcul qui se taille la part du lion [71]. Nous avons vu que plusieurs normes importantes reposent ou s'inspirent de ce formalisme. Néanmoins, ce point peut-être contesté. En effet, comme le signale Wil van der Aalst, cette filiation est parfois tellement éloignée qu'elle joue plus le rôle d'argument commercial que de véritable fondation scientifique [91].

Le pi-calcul a été imaginé par le mathématicien Robin Milner au début des années 90 [51]. Ce formalisme est utilisé pour représenter et raisonner sur des processus concurrents en interaction. Chaque processus est composé d'une ou plusieurs actions qui se succèdent séquentiellement, en parallèle, en fonction de conditions ou récursivement. Une action consiste en l'envoi ou la réception d'informations sur un canal. Ces canaux possèdent des identifiants ou noms spécifiques qui font partie des informations envoyées et reçues. Une caractéristique du pi-calcul est que ces identifiants peuvent changer, ce qui correspond à des changements de configuration. On définit un processus à l'aide d'équations exprimées dans le formalisme du pi-calcul.

3.8.2 Orc

Orc est une algèbre de processus imaginée il y a quelques années par Jayadev Misra et son équipe de l'Université du Texas. Cette algèbre repose sur quelques concepts simples.

Orc mérite sa place parmi les formalismes de représentation de workflow car en 2006, Cook, Patwardhan et Misra [19] ont proposé une implémentation des 20 motifs initiaux de van der Aalst en

Orc. Cette implémentation frappe par sa concision et son élégance ce qui suggère que Orc est un langage de représentation de workflow tout à fait viable et qui bénéficie de surcroît d'une implémentation concrète d'excellente qualité [18]. Cette implémentation est particulièrement adaptée à l'orchestration de workflows distribués dont les processus de soins constituent un cas particulier.

En Orc, le concept de base est celui de *site*. On peut considérer un site comme une procédure implémentée sur la machine hôte ou sur une machine distante. L'expression la plus simple en Orc est donc l'appel d'un site : $N(p_1, p_2 \dots p_n)$ ou N est le nom du site et $p_{1\dots n}$ représente les paramètres de cet appel. L'appel d'un site ne génère qu'une réponse au maximum et il est possible qu'un site ne réponde pas. Les appels de sites sont stricts, ce qui signifie que l'appel est lancé seulement lorsque tout ses paramètres ont une valeur. Cette propriété incite à un style de programmation proche du paradigme "dataflow" [97]. Ce mécanisme est d'ailleurs utilisé dans l'implémentation de certains motifs de workflow pour bloquer certaines évaluations.

Une implémentation concrète de Orc doit fournir une série de sites de base (ex. dans table 3.4) sans lesquels il est très difficile d'exprimer quelque chose de réellement utile. Ces sites de base sont assimilables aux fonctions natives des langages de programmation traditionnels.

Site	Explication
$let(x, y, \dots)$	Renvoie la valeur des arguments sous la forme d'un tuple.
$if(b)$	Renvoie un signal si b est vrai (true) et ne répond pas si b est faux (false).
$Signal$	Répond par un signal (réponse stéréotypée) ; équivalent à $if(true)$
$Rtimer(t)$	Retourne un signal après exactement t unités de temps

TAB. 3.4 – Liste non exhaustive de sites de base fournis par une implémentation concrète de l'algèbre Orc [39]

Les sites peuvent représenter un service, par exemple un service web : l'appel $Labo(p, d)$ peut représenter un serveur (service) de résultats d'analyses biologiques (prise de sang, urines, ...). Si p est un patient et d est une date, l'appel au service renvoie les résultats d'analyse de ce patient p valables à la date d. L'appel d'un site peut occasionner un changement de l'état interne du site appelé. Par exemple, dans ce cas ci, l'écriture d'une trace de consultation des résultats par un médecin.

Un site S peut avoir des points d'entrées multiples, ce qui se note $S.n$ où n est le nom du point d'entrée ciblé par l'appel. Cette syntaxe est volontairement proche de celle des appels de méthodes dans les langages orientés-objets.

Orc possède trois (en fait quatre mais l'alternative n'est pas strictement nécessaire) opérateurs de composition qui permettent de combiner les appels de sites entre eux (table 3.5). Voici un exemple de l'utilisation des deux premiers (composition parallèle et séquentielle) :

$$Labo1(p, d)|Labo2(p, d)) \; > x > \; email(m, x)$$

Dans cet exemple, deux sites de laboratoires d'analyses médicales sont appelés en parallèle. Dés qu'un site renvoie les résultats pour le patient p à la date d, ce résultat (x) est envoyé par e-mail par le site $email$ au médecin m. L'évaluation de cette expression peut donc déclencher l'envoi de 0 (aucun des laboratoires ne répond), 1 (un des deux sites à répondu) ou 2 (les deux répondent) e-mails par un troisième site distinct. Voici un autre exemple qui utilise l'opérateur d'élagage :

$$email(m, x) \; < x < \; (Labo1(p, d)|Labo2(p, d))$$

Nom	Notation	Description
Composition parallèle	$f \mid g$	Évaluation indépendante et en parallèle de f et de g. f et g peuvent correspondre à des appels directs de sites ou à des expressions plus complexes.
Composition séquentielle	$f > x > g$	L'expression f est évaluée. Chaque valeur publiée lors de l'évaluation de f initie une nouvelle évaluation de g et cette valeur est appelée x dans g. L'évaluation de f peut se poursuivre en même temps que plusieurs évaluations de g. Si f ne retourne aucun signal, g n'est jamais évaluée.
Composition séquentielle	$f \gg g$	Sucre syntaxique pour $f > x > g$ où x n'est pas nécessaire
Élagage	$f < x < g$	$f$et g s'exécutent en parallèle. Les partie de f dont l'exécution ne dépend pas de la valeur de x peuvent se poursuivre. Les autres doivent attendre qu'une valeur soit assignée à x. Si g publie une valeur, alors cette valeur est assignée à x, l'exécution de g se termine et les parties de f dépendantes de x peuvent s'exécuter.
Alternative	$f ; g$	$f$est tout d'abord exécutée. Si l'exécution de f s'arrête sans publier d'autre valeur, c'est g qui s'exécute.

TAB. 3.5 – Opérateurs de composition de l'algèbre Orc

Dans cet exemple, au maximum un seul e-mail contenant les résultats obtenus par le laboratoire qui aura répondu en premier sera envoyé au médecin m. Voici un dernier exemple qui utilise l'alternative :

$$(Labo1(p, d); Labo2(p, d)) > x > email(m, x)$$

Selon ce scénario, c'est seulement dans le cas ou le laboratoire n°1 serait indisponible que le laboratoire n°2 sera consulté et le résultat transmis au médecin. Le laboratoire n°1 pourrait correspondre à un laboratoire de référence et le laboratoire n°2 à un laboratoire de seconde zone.

Orc permet également l'écriture d'expressions sous la forme : $def\ f(p_1, p_2 \ldots p_n) = e$ qui définit la fonction f et ses paramètres $p_{1\ldots n}$ et dont le corps fonctionnel est l'expression e. Orc possède d'ailleurs un noyau fonctionnel assez complet qui permet la définition de fonctions, de structures de données, d'expressions conditionnelles et de correspondance de motif (pattern matching).

Le lecteur intéressé peut consulter l'article de Cook [19] pour se faire une idée de l'implémentation de l'ensemble des motifs de van der Aalst en Orc. Ce qui frappe d'emblée est la concision et l'élégance (une notion éminemment subjective) de ces implémentations.

Plusieurs sémantiques ont été données à l'algèbre Orc, notamment une sémantique temporisée sous forme de système de transitions labellisées (LTL, un type particulier de machine à état) [98]. Cette définition sémantique ouvre des possibilités de vérification des modèles exprimés en Orc à l'aide de vérificateurs de modèles (model checkers). Cette piste a déjà été explorée à l'aide du vérificateur de modèles temporels Uppaal [59].

Pour le lecteur curieux, l'implémentation concrète de Orc [18], réalisée sur la machine virtuelle Java utilise une technique avancée de manipulation du byte code [80]. Cette implémentation supporte ainsi l'instanciation de millions de fils d'exécution légers (lightweight threads) sur un seul processeur physique.

3.9 Comparaison des approches logiques et graphiques

Lu a mené en 2007 [47] une comparaison des qualités respectives des principaux langages de représentation de workflow.

Ce travail abordait ces langages essentiellement sous la perspective du flux de contrôle. La première étape a consisté à répartir les langages évalués selon leur paradigme dominant : basés sur des graphes ou basés sur des règles. Dans un second temps, cinq critères de comparaison furent définis :

1. L'expressivité, c'est à dire la capacité du langage à exprimer l'ensemble des caractéristiques d'un processus : le flux de contrôle, les données, les contraintes temporelles, ...
2. La flexibilité, c'est à dire la possibilité d'exécuter un modèle de workflow incomplet et de profiter d'éventuelles informations collectées à l'exécution pour influencer son exécution.
3. L'adaptabilité, c'est à dire la capacité pour le modèle de réagir à des circonstances exceptionnelles (non prévue au départ) lors de son exécution.
4. Le dynamisme, c'est à dire la capacité d'évolution du modèle de processus. Cette caractéristique est particulièrement importante lorsque le processus est placé dans une boucle d'amélioration continue où des modifications itératives lui sont apportées.
5. La complexité qui mesure la difficulté de modélisation, d'analyse et de déploiement du modèle de processus.

L'analyse à consisté à évaluer les langages en fonction de ces cinq critères :

- dans l'implémentation d'une partie des 20 motifs de workflow (voir section 2.6).
- dans l'implémentation d'un workflow réel d'un niveau de complexité significatif.

Les critères 2, 3 et 4 nous semblent les plus critiques lorsqu'il s'agit de modéliser des processus de soins :

- la flexibilité et l'adaptabilité, parce que la complexité des traitements, l'occurrence d'événements exceptionnels, les variations individuelles entre patients et le nombre très élevé d'options thérapeutiques rendent illusoire la conception de modèles qui tiennent compte de tous les cas de figure.
- le dynamisme parce que la science médicale progresse à grande vitesse et que la durée de vie d'un processus de soins sans modifications est en général inférieure à une année.

Du point de vue de leur support théorique, de la richesse de leur propriétés et des outils d'analyse et de vérification disponibles, les graphes et les règles logiques se valent selon cet auteur.

Les formalismes basés sur les graphes sont plus intuitifs car la correspondance entre le flux des activités du processus et les noeuds du modèle graphique est en général immédiate. Cette correspondance n'est pas aussi évidente dans les langages basés sur des règles.

En terme d'expressivité, les langages basés sur les règles ont un léger avantage, notamment dans la facilité de spécifier des contraintes temporelles sur l'exécution des tâches.

Les langages basés sur des règles ont par contre un net avantage en terme de flexibilité, d'adaptabilité et de dynamisme. Cet avantage est lié à la rigidité des modèles graphiques. Certaines approches basées sur la modification dynamique du schéma du workflow tentent d'y remédier mais sont complexes à mettre en oeuvre (voir section 3.10). Les langages basés sur des règles permettent le déploiement d'un modèle incomplet. Il est ensuite possible de le modifier en ajoutant et en supprimant des règles. Ces modifications qui correspondent à l'évolution du processus modélisé (par exemple à des fins d'amélioration) peuvent se faire sans perturber les instances du workflow en cours d'exécution. Un tel niveau de souplesse est très difficile à atteindre avec un modèle graphique.

Ces conclusions nous incitent à nous tourner vers une représentation à base de règles pour rencontrer les exigences propres aux processus médicaux.

3.10 Amélioration de la flexibilité des workflows

Plusieurs techniques et paradigmes sont envisagés pour améliorer la flexibilité des workflows :

- La manipulation directe du schéma du workflow [75]
- Une modélisation de workflow orientée-objectifs [13, **?**]
- Une modélisation de workflows orientée-agents [12, 3, 36]

Dans ce travail, c'est la solution orientée-objectifs que nous explorerons au chapitre 6. Nous nous inspirerons néanmoins de certains concepts issus du monde des agents comme la notion de plans d'activités en charge de réaliser un objectif particulier.

3.11 Résumé

Ce chapitre nous a permis de faire un tour d'horizon des trois principales familles de formalismes utilisables pour représenter des workflows : les langages basés sur la théorie des graphes, les langages basés sur la logique et les algèbres de processus. Il y est également suggéré que les approches basées sur la logique offrent sans doute certains avantages quant à la flexibilité, l'adaptabilité et le dynamisme des modèles obtenus.

La distinction entre langages basés sur les graphes et langages basés sur la logique est proposée par Lu [47] dans son article de 2007 qui est un résumé de sa thèse de doctorat écrite en 2004.

Pour établir la liste des formalismes utilisables pour représenter un workflow, l'auteur s'est principalement basé sur les publications de Oren [61] et Cicekli [17] ainsi que sur le chapitre 3 de l'ouvrage de Havey [32]. La liste des normes sous-tendues par ces différents formalismes a également été composée à partir de l'ouvrage de Havey et des pages de l'encyclopédie en ligne Wikipedia relatives à chacune d'entre-elles.

La présentation des réseaux de Petri, de leurs extensions et leur utilisation pour représenter des workflows se base sur le rapport technique publié par Oren et Haller [61]. La présentation de YAWL se base également sur le contenu de ce rapport.

La sélection des outils d'analyse présentés à la section 3.2.2 est tirée de la page suivante qui présente une liste extensive d'outils liés aux réseaux de Petri :
http ://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html (retrouvé le 4 août 2009)

La description succincte des machines à états et des diagrammes d'activités reprend certaines informations issues du chapitre 3 de l'ouvrage de Havey [32] ainsi que l'expérience de l'auteur dans ce domaine.

La description des arbres de tâches s'appuie sur l'article publié par Léon Osterweil et son équipe [62].

La description des mécanisme utilisables pour représenter un workflow à l'aide de règles est basée sur le rapport de recherche publié par Endl, Knolmayer et Pfahrer [40]

La description des systèmes orientés-agents s'appuie sur les chapitres 1 à 3 de l'ouvrage de Padgham et Winikoff [63].

La description des approches basées sur la logique temporelle a pour références :

- les pages 151 à 155 de l'ouvrage de van Lamsweerde [96]
- les pages de l'encyclopédie en ligne Wikipedia consacrée à :
  - la logique temporelle : http ://en.wikipedia.org/wiki/Temporal_logic (retrouvée le 4 août 2009)
  - la LTL : http ://en.wikipedia.org/wiki/Linear_temporal_logic (retrouvée le 4 août 2009)
  - la CTL : http ://en.wikipedia.org/wiki/Computational_tree_logic (retrouvée le 4 août 2009)
  - l'ITL : http ://en.wikipedia.org/wiki/Interval_temporal_logic (retrouvée le 4 août 2009)
- Le site internet consacré à Arthur Prior : http ://www.prior.aau.dk/index2.htm (retrouvé le 17 août 2009)

Les références à l'utilisation de la logique temporelle pour représenter et vérifier des workflows sont tirées du rapport de Oren et Haller [61]. Les références à la logique transactionnelle proviennent du même rapport.

Les deux références relatives à l'utilisation du calcul des événements pour représenter des workflows proviennent de la conclusion du chapitre 14 de l'ouvrage de Mueller [55].

La description des concepts fondamentaux du calcul situationnel est tirée de la section 3.1 de l'ouvrage de Reiter [74].

La description succincte du pi-calcul s'appuie sur la section 3.2 de l'ouvrage de Havey [32].

La description de la mécanique du langage Orc est tirée de l'article de Kitchin qui présente le language [39]. La référence à la possibilité d'implémenter les motifs de workflow en Orc est tirée de l'article de Srinivasan [80]. Enfin, certains détails relatifs à l'implémentation de Orc proviennent de Cook [18].

La comparaison entre les avantages respectifs des représentations graphiques ou logiques est due à Lu [47].

Chapitre 4

Processus de soins et systèmes de gestion de workflow

Prodiguer des soins aux patients implique de pouvoir coordonner au fil du temps les efforts d'une équipe multi-disciplinaire nombreuse, hétérogène et dispersée. Ceci est d'autant plus vrai lorsqu'il s'agit de patients chroniques atteints de multiples maladies. Ce type de patient devient de plus en plus fréquent au fur et à mesure que vieillissent nos populations.

4.1 Accroissement de la complexité médicale

La médecine, comme l'informatique, est une discipline en constante évolution. Les découvertes scientifiques et les inventions technologiques qui se succèdent rapidement remettent sans cesse en question les processus cliniques. La connaissance médicale s'accroît régulièrement et il devient difficile, même pour un spécialiste, de maîtriser les connaissances de son domaine voire celles relatives à une unique pathologie. La spécialisation médicale à outrance est le résultat de cette évolution qui accroît encore le besoin de coordination interdisciplinaire.

Les soins de santé sont donc de plus en plus complexes. En 2003, une étude multicentrique estimait qu'en Belgique, pour la pose d'une prothèse de hanche (intervention banale), un patient est confronté journalièrement à 22 contacts avec des soignants issus de 6 disciplines pour réaliser 33 actes différents, et ceci sur une durée de 10 à 12 jours [23]. Cette complexité organisationnelle a des conséquences importantes sur la sécurité des patients. La qualité des soins prodigués dépend d'une gestion réussie de cette complexité.

4.2 La crise de la qualité en médecine

Si l'on reconnaît en la majorité de soignants des personnes hautement compétentes et dévouées à leur métier, il faut aussi constater les performances médiocres du système des soins de santé. On ne compte plus les études qui relèvent le nombre important d'erreurs médicales et de soins sub-optimaux. En Europe, on estime qu'un patient hospitalisé sur dix souffre d'événements qui lui sont préjudiciables et aux conséquences éventuellement vitales [85]. Ainsi, les soins médicaux comptent parmi les activités humaines les plus risquées [45]. Ces contre-performances ont un coût humain et financier très important. A l'aube du 21ème siècle, on estimait aux États-Unis que chaque année, plus de 1,1 milliard de

dollars était consacré à éponger ces coûts [84]. On évaluait également qu'un tiers de ces coûts était attribuable à des soins inappropriés et évitables.

Ces conséquences économiques sont en partie répercutées sur les soignants qui sont contraints par les pouvoirs publics de voir plus de patients en moins de temps, de limiter le gaspillage et de faire la preuve d'une amélioration de la qualité de leurs soins.

Pendant la dernière décennie, le risque médical a été fortement investigué, particulièrement aux USA. Cette tendance est plus récente en Europe [48]. Pour faire face à ces défis, les organisations médicales s'inspirent de plus en plus de pratiques industrielles, notamment dans l'utilisation de l'informatique pour supporter leurs processus [6].

4.3 Informatisation médicale et démarche qualité

L'informatique médicale prend en charge les données liées à la pratique et à la connaissance médicale. L'usage de l'informatique est actuellement généralisé dans les organisations de santé. On constate que cet usage se déploie généralement en deux phases :

1. l'accès à l'information médicale
2. la gestion des soins selon une démarche orientée qualité

La première phase concerne l'accès à l'information médicale. Elle se matérialise sous la forme du dossier médical électronique sensé fournir au soignant les informations à propos de leurs patients au bon endroit et au bon moment. Le dossier médical électronique apporte une solution aux inconvénients liés à l'utilisation de documents papiers : piètre disponibilité, incomplétude, redondance des données et des opérations de mise à jour, erreurs, ... Cette stratégie d'accès à l'information est très utile mais néglige en général les aspects liés aux processus de soins qui nous intéressent dans ce travail. En Belgique, la plupart des organisations de soins (hôpitaux, maisons médicales, cabinets de médecine générale, ...) en sont encore à cette étape.

La seconde étape concerne la gestion des soins. Les outils mis en place doivent supporter les deux principaux types d'activités de soins : la prise de décision et la fourniture des soins proprement dit. Ces outils doivent également permettre d'inscrire ces deux types d'activités dans un cycle d'amélioration continue.

Par prise de décision, on entend les étapes par lesquelles les cliniciens déterminent quelles activités de soins vont être posées à un certain moment de l'itinéraire clinique d'un patient. Ces étapes englobent la démarche diagnostique, la sélection d'examens complémentaires, de stratégies thérapeutiques et de suivi. La résultante de ce processus décisionnel est un plan de soins supposé adéquat pour le patient. La fourniture de soins consiste à exécuter le plan de soins.

Il est intéressant de constater que le processus thérapeutique, comme certains développements informatiques, suit une démarche itérative (figure 4.1). La prise de décision produit un plan de soins qui influence lui-même l'itération décisionnelle suivante et ainsi de suite. La notion de qualité est différente si l'on est en cours de prise de décision ou si l'on fournit des soins. Pour la prise de décision, il s'agit d'identifier au bon moment les alternatives les plus pertinentes et choisir la meilleure. Dans le cas de la fourniture de soins, il s'agit de suivre le plan de soins sans commettre d'erreurs et sans gaspillage de ressources.

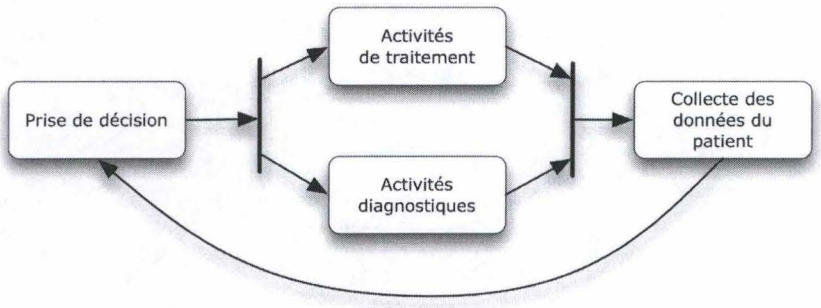

FIG. 4.1 – Vision synthétique du processus cyclique thérapeutique

Dans les soins de santé, la démarche qualité se manifeste dans plusieurs pratiques :

- la mesure de résultats et d'indicateurs de qualités
- la mise au point de bonnes pratiques de soins
- l'implémentation de ces bonnes pratiques sur le terrain

L'informatisation des soins facilite grandement la mesure. Des résultats et des indicateurs de qualités peuvent être mesurés en continu, de manière automatique sur les données collectées par les soignants.

C'est à partir de ces données que peuvent être mises-au-point des recommandations de bonnes pratiques cliniques encore appelées guidelines ou protocoles de traitement si ils se focalisent sur l'aspect thérapeutique. Une bonne partie de la recherche clinique contemporaine est consacrée à la conception de ces guidelines. On appelle cette démarche, relativement récente et qui tend à se généraliser "médecine factuelle" ou Evidence Based Medicine (EBM) [81].

Si ces guidelines sont très utiles, ils offrent néanmoins une vision idéalisée des soins, parfois très éloignée de la réalité de terrain et donc peu opérationnelle. C'est ce qui justifie la dernière étape d'implémentation. L'implémentation consiste avant tout à adapter les guidelines aux ressources et compétences disponibles dans l'organisation. C'est lors de cette étape que sont formés les soignants et que sont mises en place les procédures de renforcement de ces bonnes pratiques. A ce niveau, le processus de soins ainsi balisé porte le nom d'itinéraire clinique ou de trajet de soins (clinical pathway, care pathway). Leur informatisation est nécessaire pour pérenniser les efforts consentis à leur mise en place et leur amélioration continue.

Parce qu'ils permettent en théorie l'orchestration des processus de soins, les systèmes de gestion de workflows devraient occuper une place technologique centrale dans ces efforts d'informatisation.

4.4 Typologie des processus de soins

Il existe plusieurs approches de coordination de soins orientées processus. Elles sont en général centrées sur le patient et diffèrent par leur caractère concret ou abstrait, leur étendue (de la procédure spécifique à la prise en charge globale), leur niveau de détail, de formalisme et d'individualisation (figure 4.2).

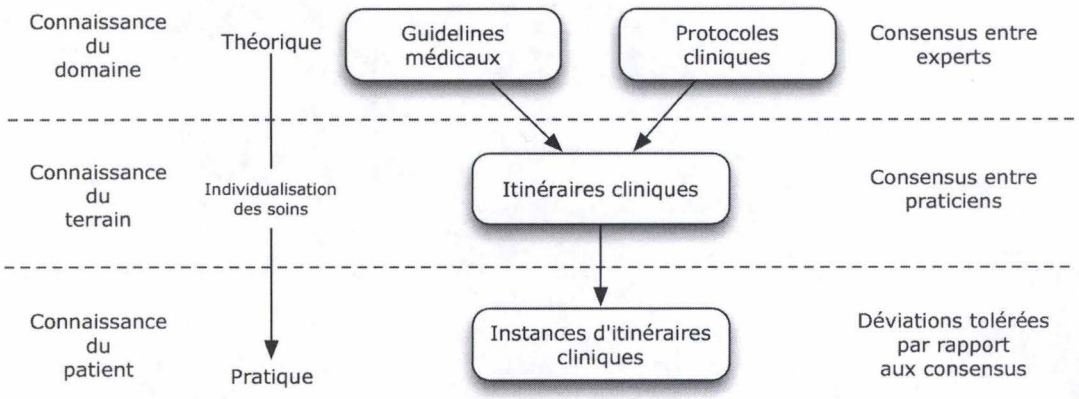

FIG. 4.2 – Typologie et niveaux d'individualisation des processus de soins

4.4.1 Les guidelines

Les guidelines médicaux sont des documents qui contiennent des recommandations quant à la prévention, au diagnostic, au traitement et au suivi d'une pathologie ou d'une classe de pathologies. Il existe des guidelines dans des domaines aussi variés que la prise en charge de l'otite moyenne aiguë, le traitement du cancer du sein ou le placement d'une prothèse totale de hanche. Ces documents sont très nombreux et balaient l'ensemble de la médecine.

Les guidelines se différencient des autres documents de recommandation par leur approche scientifique rigoureuse. Un guideline ne propose une bonne pratique que si son efficacité a été démontrée par des études cliniques dont le niveau de preuve est jugé suffisant. On appelle cette démarche rigoureuse "Evidence-Based Medicine" (EBM) ou médecine factuelle [81].

Les guidelines présentent une version abstraite et idéalisée d'une démarche de soins qui ne tient pas compte de la réalité de terrain. Suite à leur haut niveau d'exigence qualitative, ce sont en général des documents complexes qui nécessitent un effort de synthèse et d'adaptation important de la part des praticiens. Ils apportent néanmoins une aide à la décision précieuse car ils fournissent des informations liées au risque, au coût/bénéfice, et à l'efficacité des traitements. Ils présentent aussi une vision synthétique de toutes les options décisionnelles possibles. A ce titre, ils constituent une source d'information indispensable dans la définition des processus médicaux.

Des guidelines sont publiés dans la plupart des pays du monde à l'échelle nationale, voir internationale par des autorités compétentes et reconnues. En Belgique, les principaux éditeurs de guidelines sont les sociétés scientifiques de médecines. On les appelle également recommandations de bonnes pratiques cliniques ou RBP.

Il a été démontré qu'un support informatique améliore grandement l'exploitation opérationnelle des guidelines [25]. C'est pourquoi plusieurs groupes de recherche développent activement des langages de représentation de guidelines [24, 65], en fonction d'applications particulières (planification, aide à la décision, diagnostic, prescription, etc.). Dans le même temps, d'autres groupes utilisent des langages de workflow plus généralistes pour représenter ces mêmes guidelines et itinéraires cliniques [66, 56]. Les lecteurs intéressés par les langages de représentation de guidelines (il en existe une quinzaine) trouveront des références exhaustives à ce sujet sur le site internet “OpenClinical” [58].

Même si ils ont une origine distincte des langages de représentation de workflows, les langages de représentation de guidelines partagent avec eux de nombreuses caractéristiques. Ils sont notamment fondés sur les mêmes paradigmes (réseaux de Petri, règles logiques). Il est intéressant de souligner que certains de ces langages permettent d'exprimer les objectifs cliniques sous-jacents aux guidelines :

- Le langage GLIF [10] exprime les objectifs sous la forme de simples chaînes de caractères uniquement à des fins de documentation.
- En ProForma [82], les objectifs sont exprimés comme des attributs des activités. Un objectif possède deux composants : un verbe et un objet (ex. : verbe = retirer, objet = tumeur).
- Le langage Asbru est sans doute celui qui va le plus loin dans la représentation formelle des objectifs [4]. Dans ce langage, les objectifs (appelés intentions) réalisés par des plans (ensembles coordonnés d'activités) sont représentés sous forme de motifs temporels de types "achieve", "maintain" ou "avoid".

A notre connaissance, aucun de ces langages n'utilise les objectifs décrits pour influencer l'exécution du modèle.

4.4.2 Les protocoles cliniques

Les protocoles cliniques sont étroitement liés à la recherche médicale. Ce sont en quelque sorte des protocoles expérimentaux très détaillés qui fixent l'ensemble des étapes de prise en charge d'un patient lors de l'évaluation d'un nouveau traitement. Ce sont des processus de soins très documentés et complexes. Ils sont caractérisés par la description très précise des schémas de traitement en terme de contraintes temporelles strictes, de dose et d'observation de résultats d'analyses biologiques et d'effets secondaires. Ils sont également très rigides et tolèrent peu les exceptions. Un patient dont le cheminement clinique s'écarte d'un protocole en est généralement exclu.

Leur conception bénéficie d'un financement important de l'industrie pharmaceutique. Ces firmes vont jusqu'à déléguer des coordinateurs de protocoles dans les hôpitaux pour veiller à leur bon déroulement. Il est vrai que les enjeux financiers dans la recherche de nouveaux traitements sont à la hauteur des moyens mis en oeuvre.

Parmi les processus cliniques, leur caractère rigide et extrêmement détaillé font des protocoles cliniques des candidats intéressant à leur orchestration par des systèmes de gestion de workflows. Ils ne nécessitent pas autant de flexibilité dans leur exécution que d'autres types de processus de soins. Ils restent néanmoins d'un niveau de complexité bien supérieur aux workflows industriels traditionnels.

4.4.3 Les itinéraires cliniques

Un itinéraire clinique décrit un processus de soins centré sur un groupe de patients déterminé pendant une période déterminée. Il explicite les objectifs et les étapes clés de la prise en charge médicale en se basant sur des preuves scientifiques, les meilleures pratiques en vigueur et les attentes des patients. C'est un instrument de communication et de coordination de l'équipe soignante multidisciplinaire et du patient.

Les itinéraires cliniques (encore appelés "trajets de soins", "clinical pathways", "critical pathways") sont issus de méthodes de planification développées par l'industrie dans les années 50 et encore utilisées aujourd'hui. Dans les années 70 et 80, ces méthodes ont été adaptées et élargies pour répondre aux caractéristiques des soins de santé (incertitude, imprévisibilité, besoin d'individualisation et liberté thérapeutique) [106]. L'accent est mis sur la collaboration interdisciplinaire, l'Evidence-Based

Medicine [81], l'analyse des variances, et l'utilisation d'indicateurs cliniques. Les itinéraires cliniques témoignent d'une tendance vers une formalisation des processus de traitement dans de multiples secteurs de soins.

Les années 90 et 2000 ont vu la naissance d'une nouvelle approche de l'organisation hospitalière centrée sur le patient. L'itinéraire clinique devient l'axe central autour duquel les processus organisationnels prennent forme. Un programme global peut être proposé aux patients en lieu et place d'une série d'interventions distinctes et peu coordonnées. Cette période correspond à la dissémination internationale du concept d'itinéraire clinique. En 2002 est fondé le Smartgroup on Clinical Pathways (qui, en 2006, compte 786 membres issus du monde entier). En 2004 apparaît la European Pathway Association (EPA) qui lance en 2005 une vaste étude sur l'utilisation internationale des itinéraires cliniques. On y constate que 23 pays participent activement [95] : Allemagne, Angleterre, Autriche, Belgique, Danemark, Ecosse, Espagne, Estonie, Italie, Pays Bas, Suisse, Slovénie, USA, Canada, Australie, Nouvelle Zélande, Chine, Singapour, Inde, Emirats Arabes Unis, Arabie Saoudite. Pour la Belgique et les Pays-Bas, le Réseau des Itinéraires Cliniques (RIC) comptait, en janvier 2005, plus de 80 organisations membres. Des itinéraires cliniques centrés sur la prise en charge de maladies chroniques sont déployés partout en Belgique depuis avril 2009.

Actuellement, l'itinéraire clinique est devenu un véritable produit proposé par de nombreuses organisations, voire mis en vente sur Internet. Des méthodologies systématiques de conception d'itinéraires cliniques ont par ailleurs été développées [49, 94]. Elles sont en général complexes (plusieurs dizaines d'étapes), longues (de 3 à 18 mois), nécessitent un investissement conséquent et comportent des phases d'ajustement et de raffinage sur le terrain qui sont coûteuses et propices aux erreurs. Même si certaines de ces étapes peuvent être informatisées, aucun outil ne supporte encore globalement ces méthodologies. La grande majorité des itinéraires cliniques est, pour l'instant, implémentée sous forme de documents papier (textes, tableaux, organigrammes). Les systèmes de gestion de workflows ne sont actuellement pas utilisés en production dans ce domaine.

Il est intéressant de constater que la notion d'objectifs est centrale aux itinéraires cliniques. Elle fait d'ailleurs partie de sa définition. La figure 4.3 qui reproduit un formulaire de gestion d'un itinéraire clinique de prise en charge de patients palliatifs l'atteste. Les objectifs constituent des jalons naturels à atteindre dans le déroulement de l'itinéraire. Ils sont également fortement liés aux indices de performances mesurés pendant l'exécution de l'itinéraire. Ces objectifs sont, en général, des objectifs souples (soft goals). Les indices mesurés sont à rapprocher de la notion de fit criterion en ingénierie des spécifications.

4.4.4 Les workflows de service

Certains services médicaux adoptent une démarche orientée-processus pour structurer leurs activités internes. Ce sont en général des services à caractère médico-technique (radiologie, radiothérapie, médecine nucléaire, ...). Contrairement aux autres types de processus médicaux (les protocoles cliniques, les guidelines ou les itinéraires cliniques), les workflows de service sont multi-pathologiques. Il sont construits autour du flux de patients qui fréquentent le service.

4.4.5 Des processus entrecroisés

Dans une vision de l'hôpital où les services sont considérés comme des structures organisationnelles verticales, les workflows de service sont parallèles à ces structures (figure 4.4). Les itinéraires cliniques leur sont au contraire perpendiculaires. A leur intersection, ils comportent des activités communes. D'un point de vue organisationnel, les itinéraires cliniques permettent de décloisonner les services hospitaliers. Ils étendent également leurs activités au delà des frontières de l'hôpital (soins

Codes (please enter in columns) A= Achieved V=Variance (not a signature)							
Section 2	*Patient problem/focus*	*04:00*	*08:00*	*12:00*	*16:00*	*20:00*	*24:00*
Ongoing assessment ***Pain*** **Goal: Patient is pain free** • [illegible] if conscious • Pain free on movement • Appears peaceful • Consider need for positional change							
Agitation **Goal: Patient is not agitated** • [illegible] signs of delirium, terminal anguish, restlessness (thrashing, plucking, twitching) • Exclude retention of urine as cause • Consider need for positional change							
Respiratory tract secretions **Goal: Excessive secretions are not a problem** • [illegible] symptoms arise • Consider need for positional change • Symptom discussed with family/other							
Nausea & vomiting **Goal: Patient does not feel nauseous or vomits** • [illegible] if conscious							
Dyspnoea **Goal: Breathlessness is not distressing for patient** • [illegible] if conscious • Consider need for positional change.							
Other symptoms (e.g. oedema, itch)							
Treatment/procedures ***Mouth care*** **Goal: Mouth is moist and clean** • [illegible] • Mouth care assessment **at least** 4 hourly • Frequency of mouth care depends on individual need • Family/other involved in care given							
Micturition difficulties **Goal: Patient is comfortable** • [illegible] if [illegible] retention • Urinary catheter or pads, if general weakness creates incontinence							
Medication [illegible] record as N/A) **Goal: All medication is given safely & accurately** • If syringe driver in progress check [illegible] hourly according to monitoring sheet							

FIG. 4.3 – Références explicites à des objectifs dans un formulaire de gestion d'itinéraire clinique en soins palliatifs [44].

à domicile, maisons médicales, cabinets de médecins généralistes). C'est la raison pour laquelle ils constituent un excellent moyen d'amélioration de la continuité des soins.

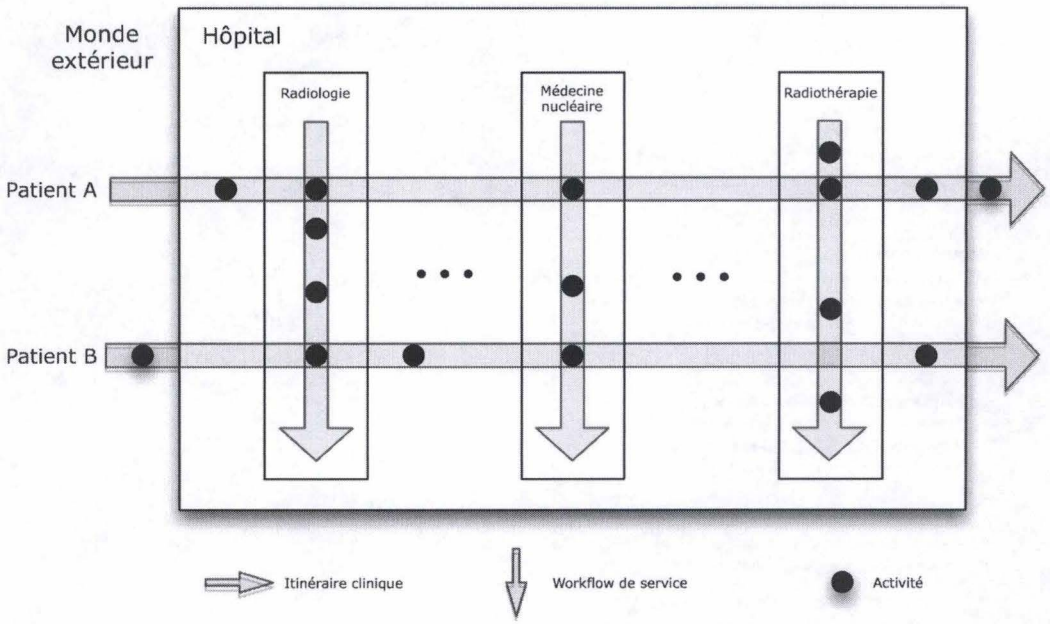

FIG. 4.4 – Relation entre workflows de services hospitaliers et itinéraires cliniques

4.5 Les systèmes de gestion de workflows dans le monde médical

Nous venons de voir qu'il existe de nombreux processus métier liés au domaine médical et plus particulièrement aux soins médicaux. Pourtant, selon l'expérience de l'auteur, on trouve peu de systèmes de gestion de workflows (SGW) déployés dans les hôpitaux pour supporter ces processus. Une explication peut-être que le monde des soins de santé offre une résistance plus élevée à l'adoption des technologies de l'information que d'autres domaines. Cette explication est plausible mais insuffisante. Une autre explication est que les SGWs actuellement commercialisés ne sont pas adaptés aux types de processus couramment rencontrés en médecine. En effet, ces processus diffèrent de la majorité des processus industriels :

- Ils sont généralement plus complexes : activités très nombreuses, décisions multiples aux nombreuses options, contraintes de temps, de dosage, de ressources à mettre en oeuvre, etc.
- Ils durent plus longtemps, particulièrement lorsqu'ils sont appliqués aux maladies chroniques pour lesquelles la prise en charge peut s'étendre sur la durée de vie complète du patient.
- Ils sont entachés d'exceptions et doivent composer avec l'incertitude permanente sur l'état du malade et son devenir.

- Ils sont géographiquement répartis (hôpital, cabinet du généraliste, domicile du patient, ...) ce qui complique leur déploiement intégré
- Ils sont multiples et s'exécutent en parallèle, de manière concurrente pour un même patient.

Cette dernière caractéristique pointe une limitation de la plupart des SGWs contemporains : ils sont "orientés-cas" [92]. C'est à dire que seule la coordination de processus logiquement indépendants est supportées. Or, lorsqu'une personne souffre simultanément de plusieurs problèmes de santé chroniques, les guidelines ou les itinéraires cliniques à suivre se multiplient. En médecine, on appelle ce problème : "la comorbidité". Il semble ne pas exister actuellement d'outil d'aide à la coordination de processus de soins qui tienne compte de ce problème de comorbidité. Les outils actuels ne tiennent pas compte des interférences possibles entre itinéraires multiples sur un même patient.

La médecine nécessite à la fois des procédures bien définies, et une grande flexibilité opérationnelle. Or, la plupart des langages de description de workflows sont purement procéduraux. Cette représentation procédurale a deux conséquences :

1. Les langages sont peu flexibles et tolèrent mal les modifications
2. Ils ne gèrent que des instances isolées de processus, sans tenir compte d'éventuelles interrelations et interférences qui peuvent exister chez un même patient.

4.6 Résumé

Les soins de santé sont confrontés à leur propre complexité et à une crise de la qualité. Ce chapitre a souligné le bénéfice potentiel lié à l'utilisation des systèmes de gestion de workflows en parallèle d'une démarche de gestion des soins orientée-processus. Néanmoins, la faible adoption actuelle de ces systèmes a également été relevée. Certaines hypothèse à ce sujet sujet ont été émises, notamment en relation avec les caractéristiques particulières des processus de soins et à l'inadéquation des systèmes disponibles aujourd'hui.

Le constat sur la complexité grandissante de la médecine, la crise de la qualité dans les soins de santé, leur informatisation en deux étapes et le rôle potentiel joué par des systèmes de gestion de processus sont suggérés au chapitre 12 de l'ouvrage de Ransom, Joshi et Nash [72].

Le reste de ce chapitre est basé sur la propre expérience de l'auteur dans ce domaine. Les études ou articles auxquels il est fait allusion sont explicitement référencés dans le texte.

4.7 Conclusion de la première partie

Nous sommes arrivés au terme de cette revue étendue du domaine des workflows et de leurs possibilités de représentation formelle. Nous avons également constaté l'intérêt potentiel d'utiliser des systèmes de gestion de workflows dans le domaine médical. Néanmoins, nous relevons leur faible adoption actuelle, sans doute due en partie à l'inadéquation des solutions industriellement disponibles.

A ce stade, il semble que pour rencontrer les exigences propres à la représentation des processus de soins, le formalisme choisi devrait :

- reposer sur la logique plus que sur la théorie des graphes.

- utiliser une représentation explicite des objectifs du processus (modèle intentionnel).
- permettre de découpler ce modèle intentionnel du modèle opérationnel (le workflow) et pouvoir modifier l'un sans modifier l'autre et inversément.
- offrir une représentation fragmentée du modèle opérationnel, assez proche du concept de plan des systèmes orientés-agents, et qui permette d'en modifier, ajouter ou supprimer des parties sans toucher au reste du modèle.
- adopter un mécanisme qui permette au modèle intentionnel d'influencer l'exécution du workflow.

Initialement, trois formalismes nous ont semblé pouvoir jouer ce rôle :

- l'algèbre de processus Orc : qui permet de représenter facilement des workflows distribués et qui dispose d'une implémentation de qualité. Néanmoins, il faut le considérer comme un langage fonctionnel. Or, il est apparu rapidement qu'une forme de raisonnement abductif serait nécessaire pour pouvoir raisonner sur le modèle d'objectifs et l'utiliser pour influencer l'exécution des workflows. Sur ce critère, Orc semble ne pas convenir.
- le calcul situationnel, utilisable pour représenter et raisonner sur des systèmes dynamiques. C'est l'absence d'outil qui permette d'effectuer un raisonnement adbuctif sur une spécification écrite en calcul situationnel qui l'a disqualifié.
- le calcul événementiel que l'on peut considérer comme quasi dual du calcul situationnel. C'est la disponibilité d'au moins deux implémentations qui supportent le raisonnement abductif et l'existence de travaux antérieurs dans le domaine des workflows qui nous ont convaincu de l'adopter.

La deuxième partie de ce travail sera donc consacrée à l'exploration de techniques orientées-objectifs de représentation de workflow avec le calcul événementiel pour médium.

Deuxième partie

Modélisation et exécution orientée-objectifs de processus de soins

Chapitre 5

Représentation logique des workflows

5.1 Le calcul des événements

Tout comme d'autres langages utilisables pour représenter des workflows (chapitre 3), le calcul des événements ou calcul événementiel (event calculus) à pour origine le domaine des bases de données. C'est Kowalski et Sergot [42] qui l'ont proposé pour la première fois au milieu des années 80. Depuis cette publication fondatrice, d'autres dialectes du calcul événementiel sont apparus pour supporter son application à d'autres domaines :

- la robotique et le planning intelligent [78]
- les workflows [16, 17]
- la sécurité des réseaux [5]
- les protocoles de communication [105]
- la compréhension du langage naturel [53]

5.2 Concepts élémentaires

Le calcul événementiel repose sur la logique du premier ordre multi-typée avec le prédicat d'égalité (voir annexe III). Ce formalisme permet de raisonner sur des systèmes dynamiques dont l'état évolue au cours du temps en fonction d'occurrences d'événements. Il repose sur trois concepts élémentaires :

- l'événement
- le fluent
- l'instant

Un événement correspond à l'occurrence d'un phénomène observable susceptible de changer l'état du système modélisé. Lorsqu'un événement survient, certaines propriétés du système qui étaient

fausses deviennent vraies et certaines propriétés qui étaient vraies deviennent fausses. Les événements servent, entre autre, à représenter l'occurence des activités dans un processus. Dans ce travail, le système correspond au processus de soins représenté et à son environnement, patient inclu.

Un fluent est une variable logique dont la valeur de vérité est susceptible de changer au cours du temps sous l'action (figure 5.1) d'événements ou pas (voir section 5.5). L'ensemble des fluents représente l'état du processus considéré à différent moments.

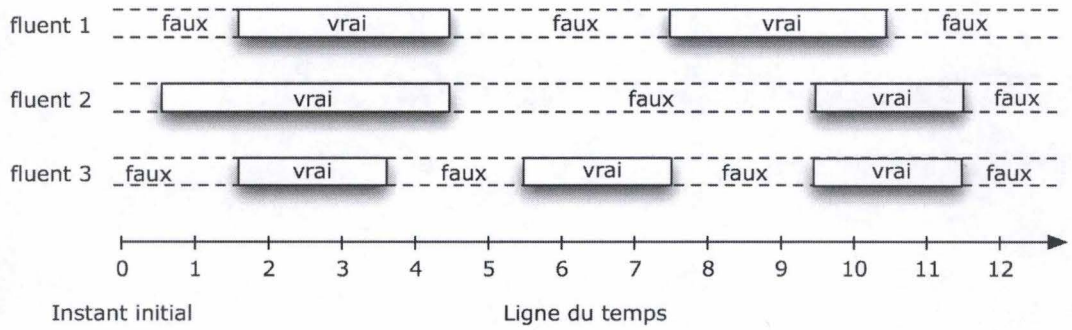

FIG. 5.1 – Un fluent est une variable logique dont la valeur de vérité est susceptible de changer au cours du temps

Un instant désigne un point particulier sur la ligne du temps. Dans ce travail, nous utilisons une représentation du temps discrète et linéaire dans laquelle chaque instant est associé à un nombre entier positif (voir section 3.7.1). Le nombre 0 représente l'instant initial du processus modélisé.

Dans le langage du calcul événementiel, ces concepts sont représentés par des types de base [55] :

- le type "événement" dont les variables sont notées e, e_1, e_2, ...
- le type "fluent" dont les variables sont notées f, f_1, f_2, ...
- le type "instant" qui est un sous type de $\mathbb{N}_0$ et dont les variables sont notées t, t_1, t_2, ...

A ces types de base se rajoutent tous les types qui correspondent aux classes d'objets dans le domaine modélisé : agent, rôle, ressource, médecin, infirmier, traitement, ...

Les fluents sont représentés par des termes. Ils peuvent donc prendre des paramètres, être arguments de prédicats et faire l'objet de quantifications.

5.2.1 Remarques sur la représentation de quantités continues

Dans certains dialectes étendus du calcul événementiel, les fluents peuvent également représenter des quantités qui varient de façon continue au cours du temps. Ce type de fluent offre des possibilités de modélisation très intéressantes dans le domaine médical. Il devient par exemple possible de représenter fidèlement les fluctuations des paramètres biologiques du patient et de construire un raisonnement sur ces fluctuations. Il est également envisageable de représenter des phénomènes pharmacocinétiques ou pharmacodynamiques pour extrapoler et raisonner sur les effets de médicaments en fonction de leurs concentrations sanguines. Ces techniques avancées ne seront pas abordées dans

ce travail. Le lecteur intéressé peut consulter le chapitre 7 de Mueller [55] consacré à la représentation du changement continu.

Bien que ces techniques avancées soient séduisantes, les implémentations informatiques concrètes du calcul événementiel les supportent mal. En outre, il est tout à fait possible de repésenter les fluctuations de ces quantités sous forme de changements discrets.

5.2.2 Remarques sur la représentation du temps

Le temps représenté par des nombres entiers positifs est suffisant pour modéliser la plupart des problèmes dans le domaine des processus de soins. La granularité de l'incrément temporel choisi sera fonction de la résolution temporelle souhaitée, elle même choisie en fonction du type de processus représenté (secondes, minutes, heures, jours, ...).

Le tableau suivant donne une idée de la résolution temporelle nécessaire en fonction du type de processus représenté :

Type de processus	Résolution temporelle nécessaire
Schémas de vaccination et de dépistage	mois - année
Suivi de pathologies chroniques (insuffisance cardiaque, rénale, diabète, hypertension, hypercholestérolémie) dont les cancers	semaine - mois
Traitements classiques dont radiothérapies et chimiothérapies	heure - jour
Soins complexes (ex. transfusion sanguine, procédures de soins intensifs)	seconde - minute
Soins incluant des machines (ex. stations d'anesthésie, respirateur, ...)	milliseconde - seconde

5.3 Les prédicats du calcul événementiel

A partir de ces trois concepts élémentaires, le calcul événementiel est construit sur des prédicats qui permettent de décrire les effets des événements sur les fluents et quels événements se produisent à quels instants. Ce travail fait référence à une version du calcul événementiel proposée par Mueller dans son ouvrage de référence [55] qui comporte 8 prédicats de base.

Le tableau suivant contient le prédicat qui permet de spécifier les occurences d'événements au cours du temps (scénarios) :

Prédicat	Description
$Happens(e, t)$	L'événement e se produit à l'instant t.

Le tableau suivant contient les prédicats qui permettent de spécifier l'effet des événements sur l'état du processus et de son environnement représenté par des fluents. Les prédicats Trajectory et Anti-Trajectory permettent de spécifier des effets en cascade.

Prédicats	Descriptions
$Initiates(e, f, t)$	Le fluent f est rendu vrai par l'événement e qui se produit à l'instant t.
$Terminates(e, f, t)$	Le fluent f est rendu faux par l'événement e qui se produit à l'instant t.
$Releases(e, f, t)$	Le fluent f est libéré de la loi d'inertie des fluents par l'événement e qui se produit à l'instant t (voir section 5.5).
$Trajectory(f_1, t_1, f_2, t_2)$	Si le fluent f_1 est rendu vrai (initié) par un événement qui se produit en t_1 et que $t_2 > 0$ alors le fluent f_2 devient vrai à l'instant $t_1 + t_2$.
$AntiTrajectory(f_1, t_1, f_2, t_2)$	Si le fluent f_1 est rendu faux (terminé) par un événement qui se produit en t_1 et que $t_2 > 0$ alors le fluent f_2 devient vrai à l'instant $t_1 + t_2$.

Enfin, le tableau suivant contient les prédicats qui permettent de spécifier l'état du processus considéré et de son environnement à n'importe quel moment :

Prédicats	Descriptions
$HoldsAt(f, t)$	Le fluent f est vrai à l'instant t, $\neg HoldsAt(f, t)$ signifie que le fluent f est faux à l'instant t.
$ReleasedAt(f, t)$	Le fluent f n'est plus sous l'influence de le loi d'inertie des fluents au temps t (voir section 5.5).

5.4 Les axiomes du calcul événementiel discret

Le calcul événementiel comporte des axiomes qui constituent la machinerie logique de support à la sémantique des prédiquats. Ces axiomes permettent de dériver des conclusions à partir d'un système décrit en terme d'événements et de fluents. Ce travail s'appuie sur une version discrète du calcul événementiel qui représente les instants par des entiers positifs [55]. Ce calcul événementiel discret comporte deux définitions et dix axiomes :

$$StoppedIn(t_1, f, t_2) \equiv \exists e, t(Happens(e, t) \wedge t_1 < t < t_2 \wedge Terminates(e, f, t)) \tag{5.1}$$

$$StartedIn(t_1, f, t_2) \equiv \exists e, t(Happens(e, t) \wedge t_1 < t < t_2 \wedge Initiates(e, f, t)) \tag{5.2}$$

La définition 5.1 énonce qu'un fluent est stoppé (rendu faux) entre les instants t_1 et t_2 si et seulement si ce fluent est terminé (rendu faux) par un événement qui survient après t_1 et avant t_2.

La définition 5.2 énonce qu'un fluent est démarré (rendu vrai) entre les instants t_1 et t_2 si et seulement si ce fluent est initié (rendu vrai) par un événement qui survient après t_1 et avant t_2.

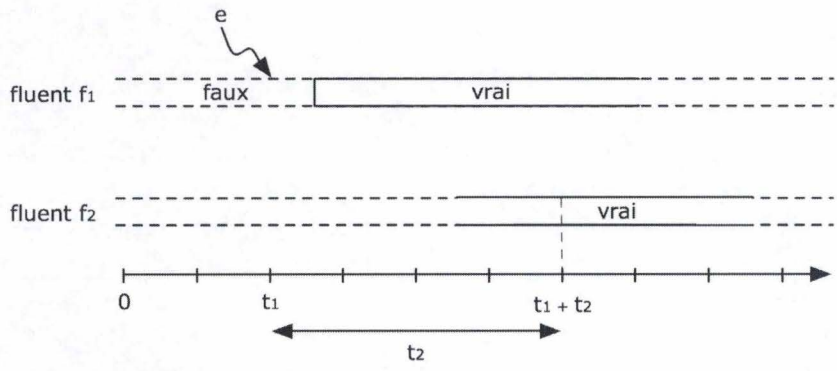

FIG. 5.2 – Le prédicat $Trajectory$ garantit que le fluent f_2 est vrai en $t_1 + t_2$ si l'événement e qui initie f_1 survient en t_1

les deux axiomes suivants spécifient la sémantique des trajectoires (trajectories) qui permettent d'exprimer que le changement de valeur de vérité d'un fluent implique qu'un autre fluent soit vrai un certain instant plus tard (figure 5.2 et 5.3). Ces axiomes rendent possible la représentation de changements d'états en cascade (effets combinés d'une même action) déclenchés par un événement unique (figures 5.2 et 5.3) :

$$(Happens(e, t_1) \wedge Initiates(e, f_1, t_1) \wedge 0 < t_2 \wedge Trajectory(f_1, t_1, f_2, t_2) \wedge \neg StoppedIn(t_1, f_1, t_1 + t_2)) \quad (5.3)$$

$$\Rightarrow HoldsAt(f_2, t_1 + t_2)$$

$$(Happens(e, t_1) \wedge Terminates(e, f_1, t_1) \wedge 0 < t_2 \wedge AntiTrajectory(f_1, t_1, f_2, t_2) \wedge \neg StartedIn(t_1, f_1, t_1 + t_2)) \quad (5.4)$$

$$\Rightarrow HoldsAt(f_2, t_1 + t_2)$$

Les deux axiomes suivants spécifient la sémantique du prédicat HoldsAt contrainte par la loi d'inertie des fluents (section 5.5) :

$$(HoldsAt(f, t) \wedge \neg ReleasedAt(f, t + 1) \wedge \neg \exists e\,(Happens(e, t) \wedge Terminates(e, f, t))) \Rightarrow HoldsAt(f, t + 1) \quad (5.5)$$

$$(\neg HoldsAt(f, t) \wedge \neg ReleasedAt(f, t + 1) \wedge \neg \exists e\,(Happens(e, t) \wedge Initiates(e, f, t))) \Rightarrow \neg HoldsAt(f, t + 1) \quad (5.6)$$

Les deux axiomes suivants spécifient la sémantique du prédicat ReleasedAt contrainte par la loi d'inertie des fluents (section 5.5). Ces axiomes énoncent qu'un fluent n'est plus soumis à la loi d'inertie des fluents tant qu'il n'est pas explicitement initié ou terminé :

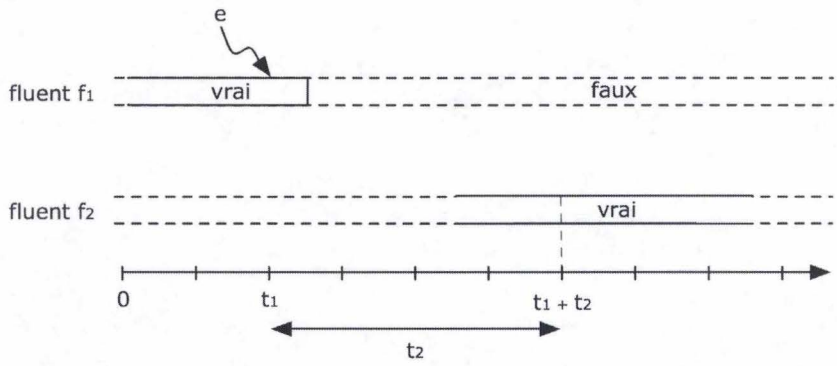

FIG. 5.3 – Le prédicat *AntiTrajectory* garantit que le fluent f_2 est vrai en $t_1 + t_2$ si l'événement e qui termine f_1 survient en t_1

$$(ReleasedAt(f, t) \wedge \neg\exists e\,(Happens(e, t) \wedge (Initiates(e, f, t) \vee Terminates(e, f, t)))) \Rightarrow ReleasedAt(f, t+1) \tag{5.7}$$

$$(\neg ReleasedAt(f, t) \wedge \neg\exists e\,(Happens(e, t) \wedge Releases(e, f, t))) \Rightarrow \neg ReleasedAt(f, t+1) \tag{5.8}$$

Enfin, les quatre derniers axiomes spécifient la sémantique de l'effet exercé par les événements sur les fluents :

$$(Happens(e, t) \wedge Initiates(e, f, t)) \Rightarrow HoldsAt(f, t+1) \tag{5.9}$$

$$(Happens(e, t) \wedge Terminates(e, f, t)) \Rightarrow \neg HoldsAt(f, t+1) \tag{5.10}$$

$$(Happens(e, t) \wedge Releases(e, f, t)) \Rightarrow ReleasedAt(f, t+1) \tag{5.11}$$

$$(Happens(e, t) \wedge (Initiates(e, f, t) \vee Terminates(e, f, t))) \Rightarrow \neg ReleasedAt(f, t+1) \tag{5.12}$$

Il découle de ces règles qu'un fluent n'est vrai qu'à l'instant qui suit l'instant d'occurence de l'événement qui l'initie. La transition de la valeur de vérité a lieu entre deux instants. Symétriquement, un fluent n'est faux qu'à l'instant qui suit l'instant d'occurence de l'événement qui le termine. L'intervalle de vérité du fluent ainsi défini est donc ouvert à gauche et fermé à droite (figure 5.4).

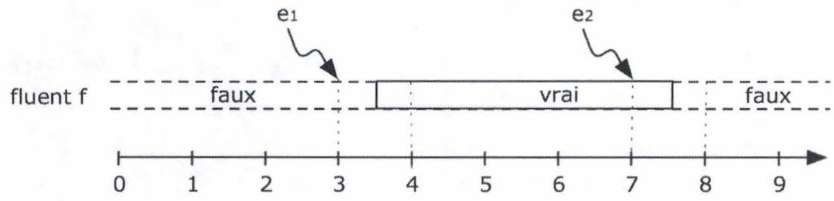

FIG. 5.4 – Décalage entre les événements d'initiation et de terminaison d'un fluent et le changement de sa valeur de vérité : $Initiates(e_1, f, 3) \wedge Terminates(e_2, f, 7)$

Ces axiomes sont présentés brièvement ici par soucs de complétude. Ils mériteraient que l'on s'attarde plus longuement sur leurs propriétés qui permettent de représenter une large variété de phénomènes : événements aux effets indirects ou non déterministes, événements composés ou concurrents, changements continus d'états, ... Le lecteur intéressé par une description plus poussée de ces axiomes, leurs effets et propriétés multiples est invité à consulter l'ouvrage de référence publié par Mueller en 2006 [55].

5.5 La loi d'inertie des fluents

La loi d'inertie des fluents découle des axiomes du calcul des événements. Cette loi énonce qu'un fluent est censé conserver sa valeur de vérité en l'absence d'occurences d'événements qui la font changer. Dans certains cas, il peutêtre utile de lever cette loi, par exemple, lorsqu'on souhaite tenir compte de la possibilité d'occurrence de circonstances exceptionnelles.

Cette possibilité de lever cette loi est particulièrement bienvenue lorsque l'on modélise des processus de soins. Par exemple, dans le traitement du cancer, un symptôme redouté des oncologues est la neutropénie fébrile. Ce symptôme signe une infection chez un patient en manque de neutrophiles (globules blancs) et donc avec des défenses naturelles diminuées. La neutropénie peut-être prévue à l'avance car elle survient dans le décours d'une chimiothérapie. La fièvre qui signe une infection est imprévisible. Au plus pouvons-nous dire que tout patient hospitalisé est à risque d'infection. C'est le problème des infections nosocomiales. En levant la loi d'inertie sur le fluent *Fébrile*, notre modèle tient compte de la possibilité de survenue inopinée de fièvre.

Voici les axiomes qui décrivent les propriétés du domaine :

$Releases(Hospitalisation(p), Fébrile(p), t)$: tout patient hospitalisé est susceptible de développer de la fièvre à n'importe quel moment.

$Initiates(Chimio(p), ChimioEnCours(p), t)$: lorsqu'une chimiothérapie démarre, elle est en cours.

$Initiates(Neutropénie(p), Neutropénique(p), t)$: une neutropénie provoque un état neutropénique chez un patient

$HoldsAt(Neutropénique(p), t) \wedge HoldsAt(Fébrile(p), t) \Rightarrow Appens(NeutropéniqueFébrile(p), t)$: une neutropénie fébrile est la combinaison d'un état de fièvre et de neutropénie.

$Initiates(NeutropéniqueFébrile(p), EnDanger(p), t)$: une neutropénie fébrile met un patient en danger

$\neg HoldsAt(ChimioEnCours(p), t-1) \wedge$

$HoldsAt(ChimioEnCours(p), t) \Rightarrow Happens(Neutropénie(p), t+7)$: la neutropénie d'un patient se déclare 7 jours après le début de sa chimiothérapie.

Et voici le scénario de l'hospitalisation du patient Dupont :

$Happens(Hospitalisation(Dupont), 0)$: Monsieur Dupont est hospitalisé au jour 0 du processus de soins.

$Happens(Chimio(Dupont), 3)$: la chimiothérapie de Monsieur Dupont débute le 3ème jour de son hospitalisation.

$\models HoldsAt(EnDanger(Dupont), 10)$: par déduction, on peut inférer que Monsieur Dupont coure un risque de neutropénie fébrile le 10ème jour de son hospitalisation.

Cette représentation permet par exemple de mettre en évidence certaines circonstances critiques lors de la vérification du modèle. Elle permet également de mettre en place des mesures d'évitement de ces circonstances et de vérifier leur efficacité. Par exemple, surveiller régulièrement la température du patient dans le décours d'une chimiothérapie et, en cas d'état subfébrile, lui donner des antibiotiques qui contrôleront le risque.

5.6 Structure d'une spécification

Une spécification en calcul événementiel comporte plusieurs éléments (figure 5.5) :

Les axiomes du calcul événementiel lui-même qui permettent d'exprimer le système modélisé (le processus et son environnement) en terme d'événements, de fluents et d'instants et de raisonner sur celui-ci. La conjonction de ces axiomes est conventionnellement notée E en calcul des événements.

Les axiomes du domaine qui décrivent la sémantique des événements, notamment en terme d'effets sur les fluents. On peut les assimiler en partie aux propriétés du domaine (section 6.2). Par exemple, les effets des traitements prodigués aux patients peuvent être spécifiés à l'aide de tels axiomes. La conjonction de ces axiomes est conventionnellement notée Σ.

Des scénarios d'événements qui décrivent ce qui se passe dans le système et à quel moment. Ces scénarios sont exprimés sous forme d'une succession d'événements temporisés. Ils sont notamment utilisés pour représenter l'occurrence du début et de la fin des activités dans un processus. La conjonction de ces formules est conventionnellement notée Δ.

L'état du système à différents moments de son évolution. Dans le cadre de la représentation d'un processus de soins, cet état comporte des informations sur le processus lui-même, le patient et son environnement. La conjonction de ces formules est conventionnellement notée Γ.

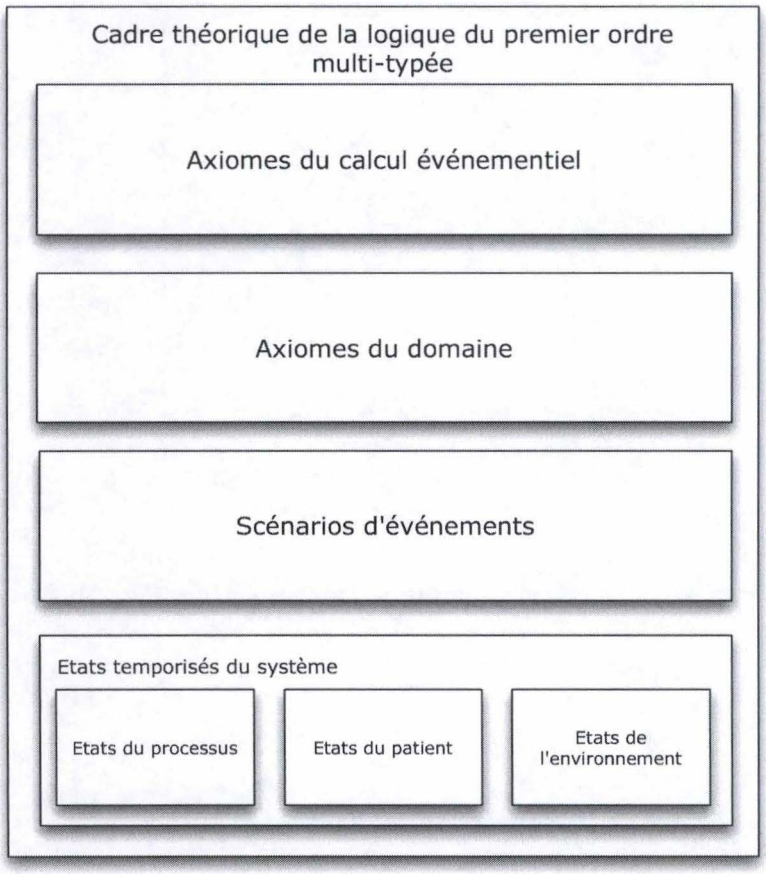

FIG. 5.5 – Eléments de la spécification d'un processus de soins à l'aide du calcul des événements

5.7 Modes de raisonnement

Plusieurs mode de raisonnement sont possibles à partir des éléments d'une spécification. :

Le raisonnement déductif : (figure 5.6) permet de faire des prédictions (projections temporelles) sur l'état du système en fonction des règles qui le régissent et des flux d'événements qui l'influencent.

Le raisonnement inductif : (figure 5.7) permet d'inférer les règles qui régissent le système. On peut assimiler ce mécanisme à une forme d'apprentissage automatique (machine learning) ou process learning [76] (section 2.4.1) dans le contexte de la modélisation de processus métier.

Le raisonnement abductif : (figure 6.23) permet à partir des règles qui régissent le système et de son état à différents moments d'inférer la succession des événements ou actions qui s'y produisent au cours du temps.

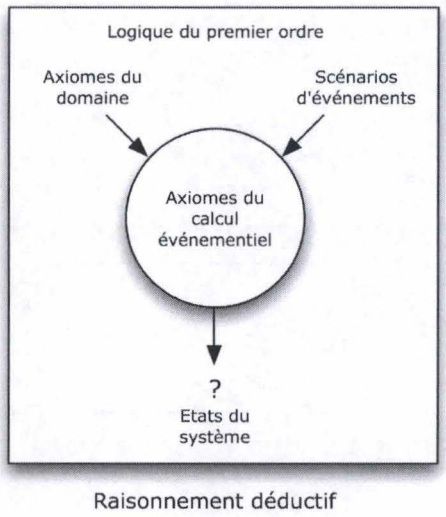

FIG. 5.6 – Représentation schématique du raisonnement déductif en calcul événementiel

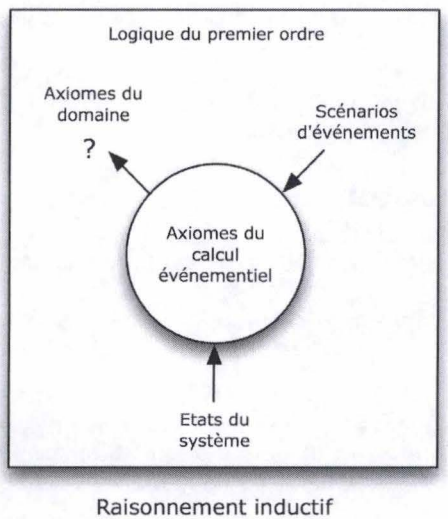

FIG. 5.7 – Représentation schématique du raisonnement inductif en calcul événementiel

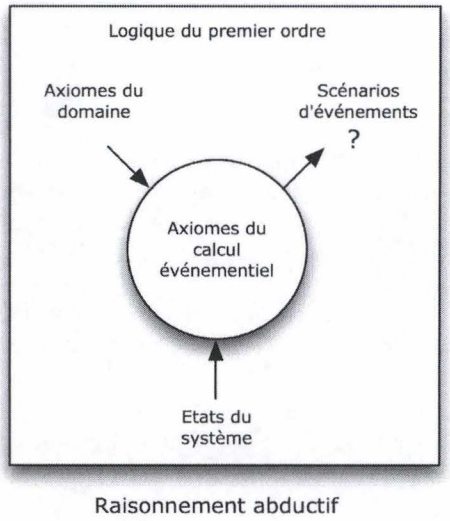

FIG. 5.8 – Représentation schématique du raisonnement abductif en calcul événementiel

Ce travail s'intéresse particulièrement au raisonnement abductif pour inférer des stratégies d'exécution de procédures de soins. L'abduction prend comme information d'entrée un modèle d'objectifs (modèle intentionnel), une représentation de l'état du patient et de l'environnement et un modèle des procédures (modèle opérationnel), le tout exprimé sous forme de formules logiques. En sortie, les axiomes du calcul des événements permettent d'inférer la succession temporisée d'événements correspondant à l'exécution des procédures.

5.8 Un exemple simple

Voici un autre exemple simple qui illustre à nouveau un processus de soins. Cette façon de représenter un processus de soins fonctionne mais n'est pas très intuitive. La section 5.11 présente une technique plus évoluée pour représenter la succession des activités d'un workflow.

Les prédicats suivants sont les axiomes du domaine (dont la conjonction est notée conventionnellement Σ) :

$Initiates(Diagnostic(p), Diagnostiqué(p), t)$: l'activité $Diagnostic$ d'un patient p fait basculer ce patient dans l'état $Diagnostiqué$.

$Initiates(Traitement(p), Guéri(p), t)$: pratiquer un traitement sur un patient le guéri
$Terminates(Traitement(p), Malade(p), t)$

$\neg HoldsAt(Diagnostiqué(p), t-1) \wedge$
$HoldsAt(Diagnostiqué(p), t) \Rightarrow Happens(Traitement(p), t)$: le traitement suit immédiatement le diagnostic.

Le prédicat suivant explicite l'état initial du patient :

$HoldsAt(Malade(Dupont), 0)$: au début du processus de soins, Monsieur Dupont est malade.

Les deux prédicats suivants explicitent le scénario de prise en charge du patient (dont la conjonction avec l'état initial de la ligne précédente est notée conventionnellement Δ) :

$Happens(Diagnostic(Dupont), 1)$: le diagnostic survient à l'instant 1

$Happens(PriseDeSang(Dupont), 2)$: une prise de sang est pratiquée à l'instant 2. Cet événement n'a pas d'effet explicité dans les axiomes du domaine.

Pour que cette spécification soit complète, il faut encore lui adjoindre les axiomes du calcul des événements dont la conjonction est conventionnellement notée E. En toute bonne logique, l'ensemble de ces informations devrait permettre d'inférer que Monsieur Dupont est guéri à l'instant 3 :

$$\mathrm{E} \wedge \Sigma \wedge \Delta \models HoldsAt(Guéri(Dupont, 3))$$

Pourtant, cette expression n'est pas valide car si les axiomes du domaine spécifient les effets de certains événements sur certains fluents, ils ne spécifient pas explicitement ce que l'ensemble des événements du modèle n'affecte pas. Par exemple, il n'est dit nulle part que l'événement PriseDeSang ne change pas la valeur de vérité du Fluent $Guéri(p)$ à un certain moment.

On appelle "problème du cadre" cette particularité liée à la représentation logique de systèmes qui évoluent dynamiquement.

5.9 Le problème du cadre

En logique des événements, si l'on représente les effets d'événements, il faut également représenter explicitement ce que les événements n'affectent pas. Cette obligation de représentation impose de fournir une quantité très importante d'informations pour effectuer un raisonnement correct.

Ce problème lié à la représentation de systèmes dynamiques avait déjà été mis en évidence à la fin des années 60. Les chercheurs en intelligence artificielle l'appellent problème du cadre (frame problem) en référence au domaine de l'animation. Dans les dessins animés réalisés à la main, ce qui ne change pas dans une scène est dessiné dans un cadre sur lequel on appose des feuilles transparentes sur lesquelles sont peints les objets mouvants.

Pour résoudre le problème du cadre, trois choses sont à expliciter dans la spécification de la section précédente :

- les non-effets implicites des événements
- la non-occurence implicite de ces même événements à chaque instant considéré
- la non-identité des événements et des fluents respectivement entre-eux

Le dernier point peut être facilement explicité à l'aide d'axiomes d'unicité des noms notés U :

$U(Malade(p), Diagnostiqué(p), Guéri(p))$
$U(Diagnostic(p), Traitement(p), PriseDeSang(p))$

Ce qui revient à dire :

$\neg(Malade(p) = Diagnostiqué(p)) \wedge \neg(Malade(p) = Guéri(p)) \wedge \ldots$
$\neg(Diagnostic(p) = Traitement(p)) \wedge \neg(Traitement(p) = PriseDeSang(p)) \wedge \ldots$ et ainsi de suite.

Ces axiomes peuvent-être automatiquement générés si l'on convient que les fluents et les événements nommés différemment ne sont pas identiques entre eux. La conjonction des axiomes d'unicité est conventionnellement notée Ω en calcul des événements.

Les deux premiers points peuvent être explicités par le mécanisme de la complétion des prédicats. Compléter un prédicat signifie indiquer précisément dans quelles circonstances il est d'application. Si dans l'exemple simple de la section précédente, on remplace les axiomes du domaine par :

$\exists p((e = Diagnostic(p) \wedge f = Diagnostiqué(p)) \vee$
$(e = Traitement(p) \wedge f = Guéri(p))) \wedge$
$((\neg HoldsAt(Diagnostiqué(p), t-1) \wedge HoldsAt(Malade(p), t)) \vee$
$((\neg HoldsAt(Guéri(p), t-1) \wedge HoldsAt(Guéri(p), t))) \Rightarrow Initiates(e, f, t))$

$\exists p(e = Traitement(p) \wedge f = Malade(p)) \wedge$
$HoldsAt(Malade(p), t-1) \wedge \neg HoldsAt(Malade(p), t) \Rightarrow Terminates(e, f, t)$

ainsi que les prédicats $Happens$ dans le scénario par :

$(e = Diagnostic(p) \wedge t = 1) \vee (e = PriseDeSang(p) \wedge t = 2) \Rightarrow Happens(e, f)$

Le problème est résolu.

L'extension d'un prédicat est l'ensemble des tuples de valeurs qui satisfont ce prédicat lorsqu'elles sont prises en arguments. Il existe un mécanisme appelé circonscription qui réalise la complétion des prédicats en leur garantissant une extension minimale. La circonscription d'une formule F qui minimise l'extension d'un prédicat p se note :

$CIRC[F; p]$

Pour résoudre le problème du cadre en calcul des événements, la circonscription doit être appliquée aux prédicats $Initiates$, $Terminates$, $Releases$ et $Happens$. Ainsi, la formule suivante est enfin vérifiée :

$$E \wedge CIRC[\Sigma; Initiates, Terminates, Releases] \wedge CIRC[\Delta; Happens] \wedge \Omega \models HoldsAt(Guéri(Dupont, 4))$$

Ce principe de circonscription est important car il justifie que l'on puisse facilement faire évoluer un modèle en lui rajoutant des axiomes. Le lecteur intéressé par les détails de ce mécanisme consultera l'article de Lifschitz publié en 1994 [46]. Les implémentations de systèmes de raisonnement basés sur calcul des événements (section 5.10) implémentent la circonscription et réalisent automatiquement la génération d'axiomes d'unicité des noms et la complétion des prédicats.

5.10 Implémentations concrètes

Il existe au moins deux implémentations concrètes qui permettent de raisonner sur des spécifications en calcul des événements :

- l'Abductive Event Calculus Planner de Murray Shanahan [79]
- le Discrete Events Reasoner (DEC) de Erik Mueller [54]

Le premier outil est un système de preuve implémenté en Prolog dont l'usage principal est le planning intelligent. Il est notamment utilisé en robotique. Son fonctionnement repose sur un raisonnement abductif pour plannifier des actions (événements) dans l'environnement modélisé.

Le second outil supporte une version discrète du calcul événementiel. Il est développé indépendamment d'un langage de programmation logique et utilise un solveur de satisfaction (SAT solver) comme moteur d'inférence. Il accepte des spécifications écrites dans un langage dont la syntaxe est proche de celle de la logique et supporte les trois formes de raisonnements décrites à la section 5.7. Ses possibilités d'utilisation dans le problème qui nous intéresse sont à priori beaucoup plus larges que le premier outil.

L'utilisation d'un SAT solver est intéressante car elle est synonyme d'efficacité dans les raisonnements. En terme de performance, ce second outil surpasse d'ailleurs largement le premier. Il utilise le format DIMACS qui permet d'exprimer de façon compacte les formules en forme normale conjonctive qui seront injectées dans le SAT solver. La plupart des SAT solvers sont compatibles avec ce format. Cette caractéristique est un gage de généricité de l'outil.

Néanmoins, après avoir utilisé cet outil pour valider les concepts présentés dans ce travail, l'auteur a été confronté à deux problèmes :

- le langage d'entrée est purement déclaratif et limité strictement à un dialecte du calcul événementiel
- la transformation de spécifications en problème SAT n'est pas garantie

Ce second problème a pour conséquence des "bloquages" de l'outil dont il est parfois difficile de déterminer la cause.

Le premier problème impose à l'analyste de travailler à un niveau d'abstraction assez bas et donc avec des modèles qui deviennent vite d'une taille très importante en terme de nombre de formules logiques.

5.11 Représentation de workflows

Deux auteurs ont appliqués le calcul événementiel à la représentation de workflows :

- Cicekli et ses collègues dans deux articles publiés respectivement en 2000 et en 2006 [16, 17].
- Mueller dans son ouvrage de référence sur le calcul des événements, également publié en 2006 [55].

L'approche présentée par Cicekli repose sur une implémentation en Prolog d'un interpréteur de workflow. Elle utilise les axiomes du calcul événementiel pour diriger l'exécution des modèles. Les possibilités procédurales de ce langage de programmation logique sont utilisées notamment pour générer des noms de variables à la volée utilisés pour la représentation des boucles et des instances multiples d'un même workflow. Cette implémentation ne supporte pas les modes de raisonnement décrits à la section 5.7.

Mueller propose une représentation très simple et purement déclarative des activités à l'aide de fluents. Elle est compatible avec les axiomes du calcul événementiel discret. Les trois modes de raisonnement décrits à la section 5.7 sont supportés ce qui ouvre des possibilités intéressantes de vérification des modèles. C'est cette approche qui est suivie dans les sections suivantes.

5.11.1 Représentation des activités

Un workflow est avant tout une séquence coordonnée d'activités (section 2.2). Nous avons vu que lors de son exécution, une activité transite par différents états d'exécution (section 2.2.3). Dans la perspective du calcul des événements, les fluents constituent une représentation naturelle des états d'une activité. Selon une approche plus simple que celle suggérée par le modèle de référence du WfMC, une activité peut être en cours (fluent $EnCours(a)$) ou terminée (fluent $Terminée(a)$). L'événement $Start(a)$ démarre l'activité a et l'événement $End(a)$ la termine.

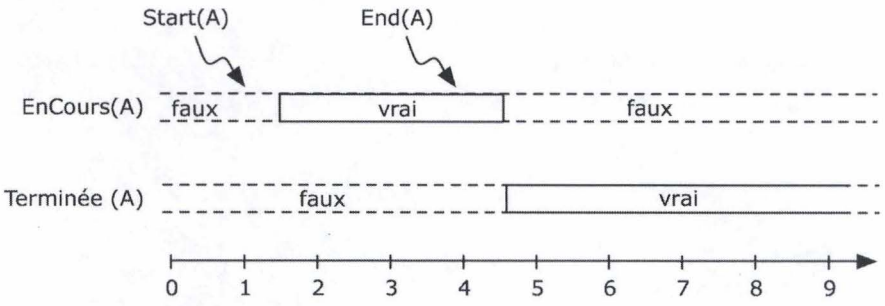

FIG. 5.9 – Représentation des états d'exécution d'une activité A à l'aide de fluents

Les axiomes du domaine suivant permettent de spécifier les transitions d'états d'une activité a en fonction des événements $Start$ et End (figure 5.9). Ces axiomes s'appuient eux-mêmes sur les axiomes et définitions du calcul événementiel discret (section 5.4). Les axiomes $Terminates$ combinés à la loi d'inertie des fluents assurent q'une activité est dans un seul état d'exécution à la fois :

$$Initiates(Start(a), EnCours(a), t)$$
$$Terminates(Start(a), Terminée(a), t)$$
$$Initiates(End(a), Terminée(a), t)$$
$$Terminates(End(a), EnCours(a), t)$$

Dans un système de gestion de workflow réel, une partie de ces événements est produite par les utilisateurs du système. Lorsqu'une activité manuelle démarre, elle est attribuée à un agent. Lorsque l'agent termine l'activité, il le signale au système par un événement End. Dans certains cas, l'agent lui-même peut décider de démarrer l'activité. Il le signale par un événement $Start$. L'occurrence de cet événement à l'instant courant peut-être ou non compatible avec le modèle du workflow. S'il est compatible, le système fait passer l'activité dans l'état $Active$. Dans le cas contraire, le système signale à l'utilisateur qu'il refuse d'acter l'événement.

5.11.2 Représentation d'une séquence d'activité

Pour rappel, le calcul événementiel repose sur une version typée de la logique du premier ordre (voir annexe : rappel de logique). Les expressions suivantes déclarent les types nécessaires :

$type$: $patient$
$type$: $médicament$
$type$: $intervention$
$instant$: t (variable)
$fluent$: $activité$
$patient$: p (variable)
$médicament$: $Cisplatine5FU$ (constante)
$intervention$: $Mastectomie$ (constante)
$activité$: a
$activité$: $Diagnostic(p)$
$activité$: $Chimiothérapie(p, Cisplatine5FU)$
$activité$: $Chirurgie(p, Mastectomie)$

FIG. 5.10 – Exemple d'une séquence d'activités (les contraintes temporelles sur la durée des tâches et des transitions ne font pas partie de la syntaxe des diagrammes d'activités UML)

Les formules suivantes décrivent la succession de trois activités de prise en charge d'un patient p pour un cancer du sein (figure 5.10). Les activités sont de durée variable. Dans ce cas de figure, l'unité temporelle choisie est le jour. La transition d'une activité à l'autre survient lorsqu'une série de pré-conditions sont rencontrées. La conjonction de ces formules est conventionnellement notée Σ (axiomes du domaine) en calcul événementiel. Cette série de formules représente le modèle du workflow :

$$\neg HoldsAt(EnCours(Chimiothérapie(p, Cisplatine5FU), t)) \wedge$$
$$\neg HoldsAt(Terminée(Diagnostic(p), t-1)) \wedge$$
$$HoldsAt(Terminée(Diagnostic(p), t) \Rightarrow$$
$$Happens(Start(Chimiothérapie(p, Cisplatine5FU), t))$$

$$\neg HoldsAt(EnCours(Chirurgie(p, Mastectomie), t)) \wedge$$
$$\neg HoldsAt(Terminée(Chimiothérapie(p, Cisplatine5FU), t-1)) \wedge$$
$$HoldsAt(Terminée(Chimiothérapie(p, Cisplatine5FU), t)) \Rightarrow$$
$$Happens(Start(Chirurgie(p, Mastectomie), t))$$

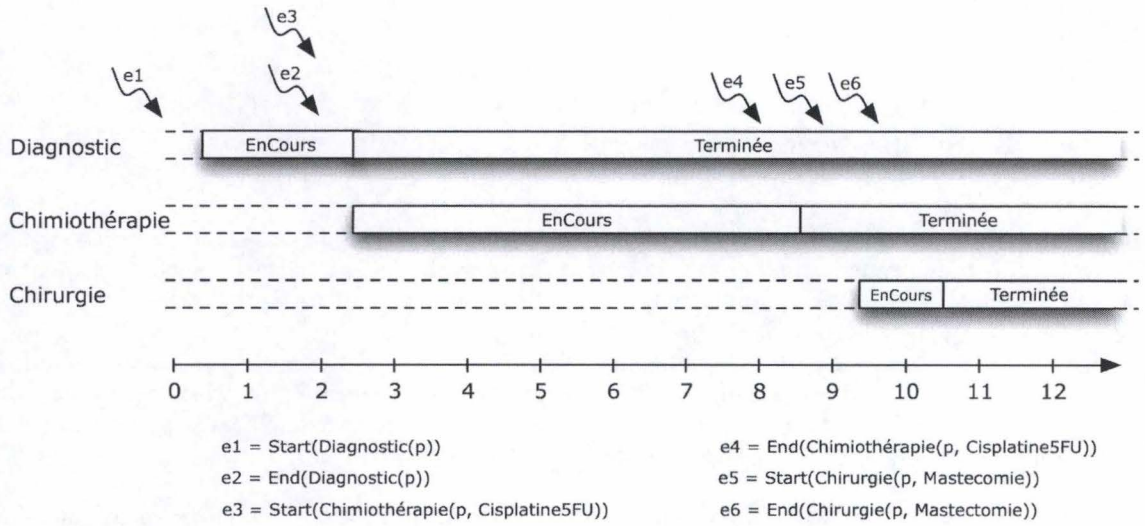

FIG. 5.11 – Représentation des états d'exécution d'activités à l'aide de fluents

La série de formules suivantes représente l'état du workflow aux différents instants considérés (dont la conjonction est conventionnellement notée Γ, états du modèle) :

$\neg HoldsAt(EnCours(a), 0) \wedge \neg HoldsAt(Completed(a), 0) \wedge$: A l'instant initial, aucune activité n'est en cours ou terminée.

$HoldsAt(EnCours(Diagnostic(p)), 1) \wedge$

$HoldsAt(EnCours(Diagnostic(p)), 2) \wedge$

$HoldsAt(Terminée(Diagnostic(p)), 3) \wedge \ldots \wedge HoldsAt(Terminée(Diagnostic(p)), n)$: n étant l'instant le plus tardif sur la période considérée.

$HoldsAt(EnCours(Chimiothérapie(p, Cisplatine5FU), 3) \wedge \ldots$
$\wedge HoldsAt(EnCours(Chimiothérapie(p, Cisplatine5FU), 8)$

$HoldsAt(Terminée(Chimiothérapie(p, Cisplatine5FU), 9) \wedge \ldots$
$\wedge HoldsAt(Terminée(Chimiothérapie(p, Cisplatine5FU), n)$: n étant l'instant le plus tardif sur la période considérée.

$HoldsAt(EnCours(Chirurgie(p, Mastectomie), 10)$

$HoldsAt(Terminée(Chirurgie(p, Mastectomie), 11) \wedge \ldots$
$\wedge HoldsAt(Terminée(Chirurgie(p, Mastectomie), n)$: n étant l'instant le plus tardif sur la période considérée.

Enfin, cette dernière série de formules représente le scénario d'occurrence des activités au cours du temps (dont la conjonction est conventionnellement notée Δ, scénarios d'occurence d'événements) :

$Happens(Start(Diagnostic(p)), 0)$
$Happens(End(Diagnostic(p)), 2)$
$Happens(Start(Chimiothérapie(p, Cisplatine5FU)), 2)$
$Happens(End(Chimiothérapie(p, Cisplatine5FU)), 8)$
$Happens(Start(Chirurgie(p, Mastectomie)), 9)$
$Happens(Terminée(Chirurgie(p, Mastectomie)), 10)$

Ainsi, le modèle est complet. Dans la réalité, il n'est pas nécessaire de travailler avec des modèles complets puisque les parties manquantes peuvent être reconstituées par raisonnement. Nous avons vu à la section 5.7 que le calcul événementiel supporte 3 types principaux de raisonnements.

Dans cet exemple, le raisonnement déductif permet d'inférer l'état du workflow (les activités en cours ou terminées, exprimées sous forme de fluents) à partir d'une liste d'occurrence d'événements *Start* et *End*. Ces occurrences d'événements correspondent à la trace d'exécution du workflow. Ce genre de trace peut-être récupérée dans les log files des systèmes d'informations médicaux. Le raisonnement déductif peut-être appliqué à ce type d'information pour donner une vision :

rétrospective du workflow à un moment de son exécution : "lorsque la résonnance magnétique nucléaire à été pratiquée le 23 janvier 2009, le patient était dans sa troisième cure de chimiothérapie (FEC + Taxotère). Il était également traité par Novaban, Litican et Neupogen".

actuelle : "aujourd'hui, le patient est au troisième jour de sa sixième cure de chimiothérapie (FEC + Taxotère). Il est également sous Paracétamol 500mg 4X/j et Codéine 30mg 4X/j".

prospective : "le 15 avril 2009, 2 cas de figures sont possibles : soit le patient sera au 6ème jour de sa 8ème cure, soit le traitement aura été suspendu pour effets secondaires non-maîtrisés".

Ce dernier point illustre le fait que lorsqu'il existe plusieurs modèles compatibles avec la spécification, le moteur d'inférence les renvoie tous. Le raisonnement déductif peut se noter de la façon suivante :

$$\mathrm{E}\wedge CIRC[\Sigma;\ Initiates,\ Terminates,\ Releases] \wedge CIRC[\Delta;\ Happens] \wedge \Gamma \wedge \Omega \models \Gamma'$$

La présence de Γ à gauche rend compte du fait que certaines informations sur l'état du processus peuvent se trouver dans la spécification fournie au moteur d'inférence pour le contraindre à considérer un ou plusieurs cas particuliers. Dans notre exemple, si nous ne fournissons qu'une séquence incomplète de prédicat *Happens*, le moteur d'inférence va sortir plusieurs successions d'états possibles. On peut alors imposer quelques états intermédiaires pour en limiter le nombre.

Le raisonnement abductif peut être également pratiqué. A partir d'une spécification fournie en entrée, il permet d'inférer des successions d'événements (scénarios) qui mènent le workflow d'un état initial à un état final, en passant éventuellement par plusieurs états intermédiaires imposés. Dans notre exemple, si l'état initial :

$$\neg HoldsAt(Active(a), 0) \wedge \neg HoldsAt(Completed(a), 0)$$

et l'état final :

$HoldsAt(Completed(a), n)$: n étant l'instant le plus tardif sur la période considérée.

sont fournis et que la spécification ne contient pas de scénarios d'événements (succession de prédicats *Happens*), le moteur d'inférence retourne les scénarios (traces d'exécution) qui permettent d'atteindre cet état final. Dans cet exemple, il n'y en a bien entendu qu'un.

La vision prospective offerte par le raisonnement abductif est d'un intérêt particulier puisqu'elle correspond à la plannification des futures activités du workflow. Nous exploiterons cette propriété dans le chapitre suivant.

Le raisonnement déductif est plus délicat à mettre en oeuvre car les modèles inférés sont parfois difficiles à interpréter en terme de workflows. Pour rappel, à partir des axiomes du calcul événementiel, de scénarios d'événements et d'états intermédiaires du processus, il infère les axiomes du domaine. Nous ne nous y attarderons pas.

5.11.3 Représentation des motifs de division et dē synchronisation parallèle

Ce fragment de workflow (figure 5.12) décrit les activités diagnostiques qui se déroulent entre la consultation initiale où la suspicion clinique d'un cancer du sein est posée et le premier staff multidisciplinaire où les décisions thérapeutiques qui concernent la patiente sont prises. Ces activités diagnostiques comportent une résonnance magnétique nucléaire (RMN), une mammographie des deux seins et une biopsie de la tumeur incriminée. Ces activités peuvent se dérouler dans un ordre indifférent. Dans cette spécification, elles ont toutes une durée de 1 (jour).

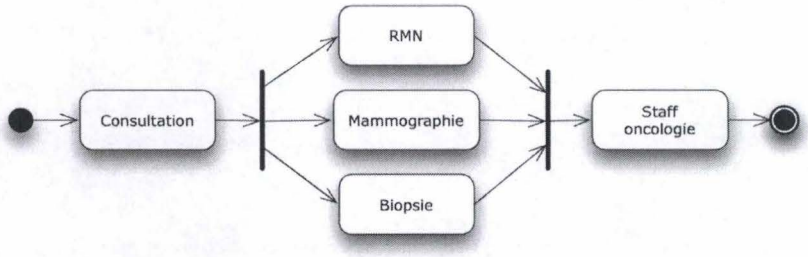

FIG. 5.12 – Activités diagnostiques parallèles dans un cancer du sein

Seuls les axiomes qui décrivent le modèle du workflow sont présentés. Il ne s'agit donc pas d'une spécification complète comme présentée à la section précédente. Après les déclarations de types, les trois premiers groupes de formules décrivent la division parallèle. Le dernier décrit la synchronisation :

$type : agent$
$instant : t$
$fluent : activité$
$agent : patient$
$agent : médecin$
$patient : p$
$médecin : m$

$activité : Consultation(p, m)$
$activité : RMN(p)$
$activité : Mammographie(p)$
$activité : Biopsie(p)$
$activité : Staff(p)$

$\neg HoldsAt(EnCours(RMN(p)), t) \wedge$
$\neg HoldsAt(Terminée(Consultation(p, m)), t-1) \wedge$
$HoldsAt(Terminée(Consultation(p, m)), t) \Rightarrow$
$Happens(Start(RMN(p), t)$

$\neg HoldsAt(EnCours(Mammographie(p)), t) \wedge$
$\neg HoldsAt(Terminée(Consultation(p, m)), t-1) \wedge$
$HoldsAt(Terminée(Consultation(p, m)), t) \Rightarrow$
$Happens(Start(Mammographie(p), t)$

$\neg HoldsAt(EnCours(Biopsie(p)), t) \wedge$
$\neg HoldsAt(Terminée(Consultation(p, m)), t-1) \wedge$
$HoldsAt(Terminée(Consultation(p, m)), t) \Rightarrow$
$Happens(Start(Biopsie(p), t)$

$\neg HoldsAt(EnCours(Staff(p), t) \wedge$
$((\neg HoldsAt(Terminée(RMN(p)), t) \wedge HoldsAt(Terminée(RMN(p)), t)) \vee$
$((\neg HoldsAt(Terminée(Mammographie(p)), t) \wedge HoldsAt(Terminée(Mammographie(p)), t)) \vee$
$((\neg HoldsAt(Terminée(Biopsie(p)), t) \wedge HoldsAt(Terminée(Biopsie(p)), t)) \wedge$
$HoldsAt(Terminée(RMN(p)), t) \wedge$
$HoldsAt(Terminée(Mammographie(p)), t) \wedge$
$HoldsAt(Terminée(Biopsie(p)), t) \Rightarrow$
$Happens(Start(Staff(p)), t)$

5.11.4 Représentation des motifs de choix et de synchronisation exclusive

Ce fragment de workflow (figure 5.13) représente une version simplifiée du processus décisionnel suivi pour définir l'approche thérapeutique chez une patiente atteinte d'un cancer du sein.

La ligne thérapeutique est définie lors d'un staff multi-disciplinaire d'oncologie pendant lequel un des traitements pré-chirurgicaux possibles est choisi. Cette décision repose principalement sur l'extension prise par la tumeur. T1-2 correspond à une tumeur localisée qui sera traitée par chimiothérapie néo-adjuvante. T3 désigne une tumeur invasive qui sera traitée de façon combinée par radiothérapie et chimiothérapie néo-adjuvantes. T4 correspond à un cancer très invasif qui s'est déjà disséminé à distance. Pour ce type de cancer, un traitement qui prend également les métastases en compte est nécessaire. Seuls les axiomes qui décrivent le modèle du workflow sont présentés. Les gardes placées sur les transitions qui mènent aux différentes options thérapeutiques sont liées à la valeur de vérité de trois fluents distincts. Ces fluents sont de la forme $Stadification(patient, organe, stade)$ où $organe$ correspond à l'organe atteint et $stade$ à un code de stadification TNM international (simplifié dans cet exemple). Par concision, les types ne sont plus déclarés :

$\neg HoldsAt(EnCours(Chimiothérapie(patient)), t) \wedge$
$\neg HoldsAt(Terminée(StaffOncologie(patient)), t-1) \wedge$

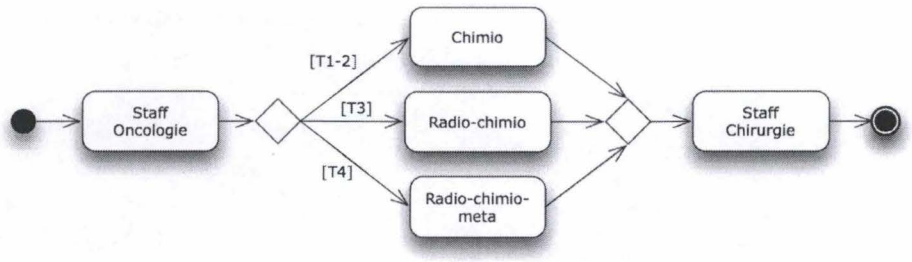

FIG. 5.13 – Sélection d'approches thérapeutiques dans un cancer du sein

$HoldsAt(Terminée(StaffOncologie(patient)), t) \wedge$
$(HoldsAt(StadificationCancer(patient, Sein, T1), t) \vee$
$(HoldsAt(StadificationCancer(patient, Sein, T2), t)) \Rightarrow$
$happens(Start(Chimiothérapie(patient)), t)$

$\neg HoldsAt(EnCours(RadioChimiothérapie(patient)), t) \wedge$
$\neg HoldsAt(Terminée(StaffOncologie(patient)), t-1) \wedge$
$HoldsAt(Terminée(StaffOncologie(patient)), t) \wedge$
$HoldsAt(StadificationCancer(patient, Sein, T3), t) \Rightarrow$
$happens(Start(RadioChimiothérapie(patient)), t)$

$\neg HoldsAt(EnCours(RadioChimiothérapieMétastatique(patient)), t) \wedge$
$\neg HoldsAt(Terminée(StaffOncologie(patient)), t-1) \wedge$
$HoldsAt(Terminée(StaffOncologie(patient)), t) \wedge$
$HoldsAt(StadificationCancer(patient, Sein, T4), t) \Rightarrow$
$happens(Start(RadioChimiothérapieMétastatique(patient)), t)$

$\neg HoldsAt(EnCours(StaffChirurgie(patient), t) \wedge$
$((\neg HoldsAt(Terminée(Chimiothérapie(patient)), t-1) \wedge$
$HoldsAt(Terminée(Chimiothérapie(patient)), t)) \vee$
$((\neg HoldsAt(Terminée(RadioChimiothérapie(patient)), t-1) \wedge$
$HoldsAt(Terminée(RadioChimiothérapie(patient)), t)) \vee$
$((\neg HoldsAt(Terminée(RadioChimiothérapieMétastatique(patient)), t-1) \wedge$
$HoldsAt(Terminée(RadioChimiothérapieMétastatique(patient)), t))) \Rightarrow$
$Happens(Start(StaffChirurgie(patient)), t)$

5.11.5 Représentation d'une itération conditionnelle

Les axiomes suivants représentent une itération conditionnelle sur un bloc d'activités. A la fin de chaque cure de chimiothérapie, la patiente est vue en consultation par le médecin. La décision de

passer à la cure suivante est prise en fonction de son état général. En cas d'interruption ou si le schéma chimiothérapeutique prévu est accompli, un bilan d'extension est réalisé pour évaluer les résultats du traitement.

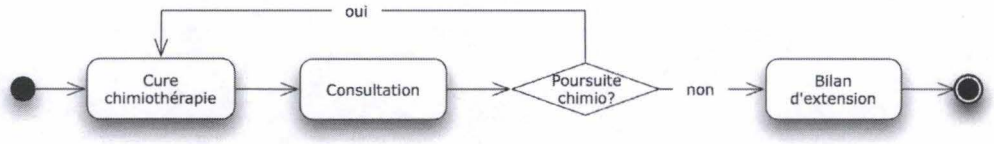

FIG. 5.14 – Itération conditionnelle de cures de chimiothérapie

$\neg HoldsAt(EnCours(Consultation(patient)), t) \wedge$
$\neg HoldsAt(Terminée(CureChimio(patient, schéma)), t-1) \wedge$
$HoldsAt(Terminée(CureChimio(patient, schéma)), t) \Rightarrow$
$Happens(Start(Consultation(patient)), t)$

$\neg HoldsAt(EnCours(CureChimio(patient, schéma)), t) \wedge$
$\neg HoldsAt(Terminée(Consultation(patient)), t-1) \wedge$
$HoldsAt(Terminée(Consultation(patient)), t) \wedge$
$HoldsAt(PoursuiteChimio(patient), t) \Rightarrow$
$Happens(Start(CureChimio(patient, schéma)), t)$

$\neg HoldsAt(EnCours(CureChimio(patient, schéma)), t) \wedge$
$\neg HoldsAt(Terminée(Consultation(patient)), t-1) \wedge$
$HoldsAt(Terminée(Consultation(patient)), t) \wedge$
$\neg HoldsAt(PoursuiteChimio(patient), t) \Rightarrow$
$Happens(Start(BilanExtension(patient)), t)$

Ces motifs de contrôle élémentaires suffisent à représenter une majorité de workflows. Néanmoins, si le besoin s'en fait sentir, il est a priori possible de représenter l'ensemble des motifs étendus décrits à la section 2.6 à l'aide du calcul des événements.

5.12 Limites de cette approche

Cette forme de représentation connaît certaines limites d'expressivité. Par exemple, dans la boucle de la section précédente, on considère que ce sont les mêmes activités *CureChimiothérapie*(*patient*, *Schéma*) et *Consultation*(*patient*) qui sont répétées plusieurs fois. Dans certains cas, on souhaiterait qu'il s'agisse chaque fois d'une nouvelle instance de ces activités. C'est notamment nécessaire pour pouvoir se référer à une cure ou une consultation particulière : "pendant la troisième cure de FEC de Madame Durant", "lors de la consultation du 23 janvier 2007 de Madame Dupont". Dans ce cas, chacune de ces instances doit-être explicitement représentée par des axiomes dédiés.

Les modèles de workflows présentés aux sections précédentes ne considèrent que deux états d'exécution possibles pour les activités. Pour se rapprocher du modèle de référence de la WfMC, on peut considérer deux états d'exécution d'activité (*Interrompue* et *Annulée*) ainsi que deux événements (*Pause*, *Stop*) supplémentaires (figure 5.15).

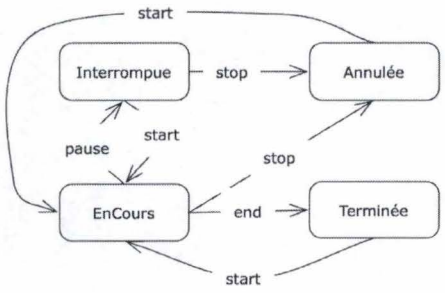

FIG. 5.15 – Transitions de valeur de fluents pour représenter une activité qu'on peut interrompre et annuler.

Les axiomes suivants garantissent ces transitions d'états :

$Initiates(Start(a), EnCours(a), t)$
$Terminates(Start(a), Interrompue(a), t)$
$Terminates(Start(a), Annulée(a), t)$
$Terminates(Start(a), Terminée(a), t)$
$Initiates(Pause(a), Interrompue(a), t)$
$Terminates(Pause(a), EnCours(a), t)$
$Initiates(Stop(a), Annulée(a), t)$
$Terminates(Stop(a), EnCours(a), t)$
$Terminates(Stop(a), Interrompue(a), t)$
$Initiates(End(a), Terminée(a), t)$
$Terminates(End(a), EnCours(a), t)$

Les formules suivantes représentent une activité A capable d'être interrompue et annulée :

$\neg HoldsAt(Interrompue(A), t) \wedge$
$HoldsAt(EnCours(A), t) \Rightarrow$
$Happens(Pause(A), t)$

$\neg HoldsAt(EnCours(A), t) \wedge$
$HoldsAt(Interrompue(A), t) \Rightarrow$
$Happens(Start(A), t)$

$\neg HoldsAt(Annulée(A), t) \wedge$
$HoldsAt(Interrompue(A), t) \Rightarrow$
$Happens(Stop(A), t)$

$\neg HoldsAt(Annulée(A), t) \wedge$
$HoldsAt(EnCours(A), t) \Rightarrow$
$Happens(Stop(A), t)$

$\neg HoldsAt(EnCours(b), t) \wedge$
$\neg HoldsAt(Terminée(A), t-1) \wedge$
$HoldsAt(Terminée(A), t) \Rightarrow$
$Happens(Start(b), t)$

$\neg HoldsAt(EnCours(b), t) \wedge$
$\neg HoldsAt(Annulée(A), t-1) \wedge$
$HoldsAt(Annulée(A), t) \Rightarrow$
$Happens(Start(b), t)$

Les modèles deviennent ainsi extrêmement fastidieux à construire. Ces limites d'expressivité découlent de la nature purement déclarative du calcul événementiel. L'implémentation de Shanahan, évoquée plus tôt dans ce chapitre est construite sur un interpréteur Prolog. Elle permet d'inclure certains éléments procéduraux propres à ce langage dans les spécifications. L'implémentation de Mueller, utilisée par l'auteur pour valider les concepts présentés dans ce travail et qui offre des possibilités plus riches de raisonnement, ne supporte pas ces éléments procéduraux.

Ces exemples mettent en lumière la nécessité de disposer d'un langage de plus haut niveau, idéalement graphique, pour représenter la succession des activités. Les diagrammes d'activités UML ou la notation BPMN sont deux bons candidats potentiels. Il est ensuite envisageable de transformer automatiquement ces modèles en formules logiques correspondantes.

5.12.1 Remarques

Un des objectifs de ce travail est de démontrer que la combinaison d'un modèle intentionnel et d'un modèle opérationnel permet d'obtenir des workflows plus flexibles et avec une meilleure capacité d'adaptation. Dans cette représentation, les aspects intentionnels (modèle d'objectifs) sont clairement séparés des aspects opérationnels (ensemble de workflows autonomes de petites taille). Nous venons de voir que le calcul événementiel offre la possibilité de représenter ce deuxième aspect. Dans un chapitre ultérieur, nous verrons qu'il permet également de représenter le premier. Le recours à un seul formalisme est bien évidemment avantageux. Néanmoins, le recours au calcul événementiel pour représenter les détails procéduraux des plans est facultatif. Tout autre formalisme (chapitre 3) peut à priori convenir à condition que les pré- et post-conditions du plan modélisés soient exprimées en calcul événementiel pour permettre l'inférence de stratégies d'exécution.

Une autre remarque est que les activités sont susceptibles d'avoir certains effets sur l'état du patient ou de son environnement. Or, dans ce chapitre, nous avons peu représenté ces effets à part dans les quelques exemples simples du début. Il ne s'agit pas d'un oubli et nous donnerons d'autres exemples dans le chapitre suivant.

5.12.2 Conventions de nommage particulières

La représentation de l'exécution parallèle de plusieurs instances d'un même workflow se heurte au problème de la différenciation des mêmes activités exécutées dans ces instances différentes. Ce problème n'est pas à négliger dans le domaine des soins. Nous avons évoqué à la section 4.5 le problème de la comorbidité qui pointe le fait que de plus en plus de patients souffrent de pathologies chroniques multiples et intriquées. Il n'est donc pas rare qu'un patient suive en même temps deux itinéraires cliniques parallèles. La confusion entre activités du même nom (ex. *Consultation*) est plus que probable.

Une manière de procéder est de faire en sorte que chaque instance de workflow possède un identifiant unique. Cet identifiant est associé aux noms des instances d'activité : $EnCours(w, a)$ ou w est l'identifiant du workflow et a l'identifiant de l'activité.

Ce problème d'unicité de nom se pose également dans les boucles. A chaque itération, c'est une nouvelle instance d'activité qui est créée. Ces instances doivent être identifiées non seulement par l'identifiant de l'instance courante du workflow mais également par un numéro d'itération. Il faut également identifier de façon unique les instances d'événements.

Nous sommes bien conscients de ces impératifs de nommage. Néanmoins, pour des raisons de simplicité et de concision, nous n'avons en général pas adopté ces conventions dans les fragments de modèles qui illustrent ce texte.

5.13 Projection des fluents sur une ligne du temps

De part leur nature, les fluents se prêtent particulièrement bien à la projection sur une ligne du temps. On peut exploiter cette propriété pour générer une représentation visuelle attractive d'un processus sous la forme d'une ligne du temps. La figure 5.16 est une capture d'écran de l'outil évoqué dans l'introduction de ce travail et au développement duquel l'auteur participe. Cette représentation est générée à partir de données obtenues dans le dossier médical informatisé du patient et ne se base donc pas sur un modèle de fluent. Néanmoins, un modèle de processus de soins spécifié en calcul des événements pourrait tout à fait être visualisé de la même façon.

Les modes de raisonnement déductif et inductif pourrait-être appliqués pour générer des séquences d'états et d'événements liés au patient, à son environnement ou au processus de soins lui-même. Ces séquences seraient prospectives ou rétrospectives en fonction d'un instant présent choisi arbitrairement. Un tri, basé par exemple sur des conventions de nommage des événements et des fluents, permettrait de n'afficher que les informations concernant les traitement, les état pathologiques du patient, les occurrences d'actes de soins, l'utilisation de ressources, ...

5.14 Résumé

Ce chapitre a présenté les concepts élémentaires, les prédicats et les axiomes du calcul événementiel. Les différents éléments d'une spécification exprimée dans ce formalise y ont été détaillés. A partir de ces spécifications, trois principaux modes de raisonnement sont possibles : le raisonnement déductif, inductif et abductif. Ce dernier occupe une place particulière dans ce travail. Le problème du cadre a également été abordé. Ce problème se pose lors de la représentation de systèmes dynamiques en évolution tels des processus. L'existence de ce problème implique que les systèmes de raisonnement automatiques qui implémentent le calcul des événements doivent comporter des mécanismes de génération d'axiomes d'unicité et de complétion des prédicats. Enfin, ce chapitre a démontré qu'il était possible de représenter précisément l'enchaînement coordonné des activités d'un workflow à l'aide de formules logiques. Les limites de cette approche notamment en terme de longueur des spécifications à produire ont également été pointées. Il semble nécessaire de disposer d'outils capables de générer ce type de spécifications à partir de formalismes de plus haut niveau, idéalement graphiques.

L'ouvrage de Mueller [55] consacré au calcul événementiel et à ses applications constitue la principale source d'information de ce chapitre :

- Les références aux différents domaines d'application du calcul des événements sont tirées de l'introduction et des notes bibliographiques du chapitre 14

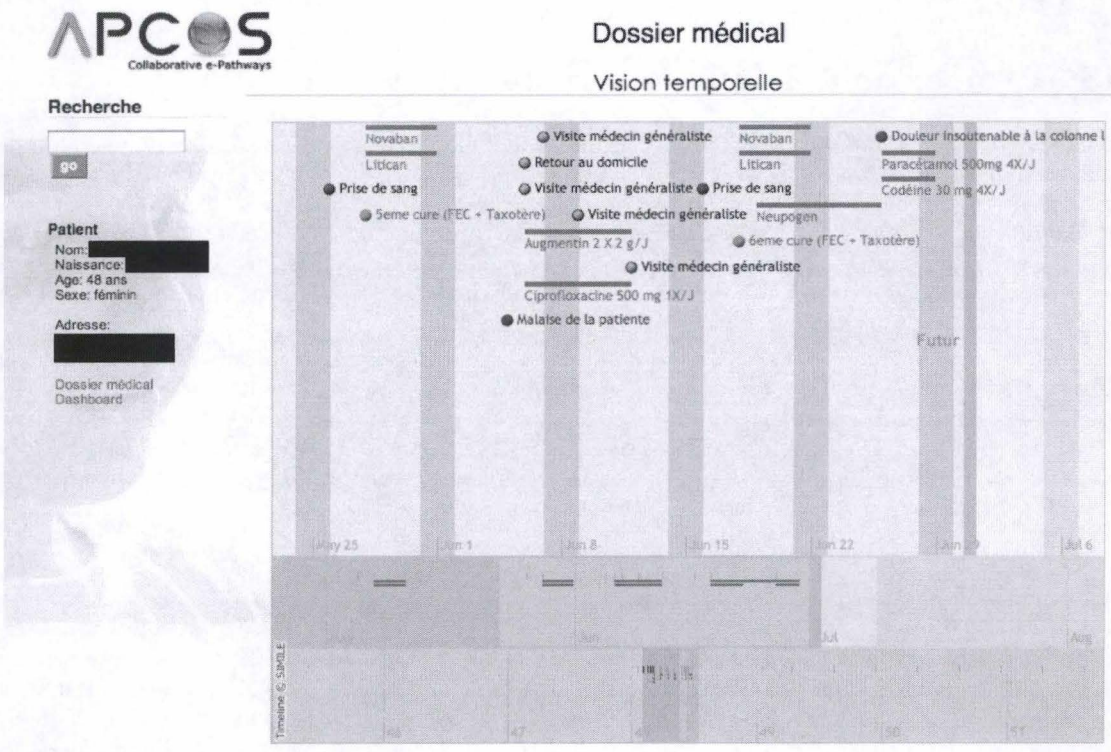

FIG. 5.16 – Représentation des états successifs d'un patient et de certains événements (points) sur une ligne du temps

- La proposition de représenter par des fluents les états d'exécution des activités d'un workflow ainsi que la technique d'expression des motifs de contrôle trouve sa source à la section 14.1.2 de ce même chapitre.
- Les prédicats et axiomes du calcul événementiel discret sont posés par Mueller à la section 2.3.2 du chapitre 2.
- La présentation des modes de raisonnements possibles est basée sur la section 2.8 de ce même chapitre.
- La discussion sur le problème du cadre et sa résolution se base sur les informations dans les sections 2.4.1 à 2.7.1. L'étymologie de l'appellation "problème du cadre" provient de la page de l'encyclopédie en ligne wikipedia qui lui est consacrée :
  http ://en.wikipedia.org/wiki/Frame_problem (retrouvé le 4 août 2009).

Chapitre 6

Conception de workflows orientée-objectif

Nous avons constaté que la plupart des paradigmes de modélisation de workflows s'intéressaient principalement à la représentation du flux de contrôle en terme de succession d'activités. Or, à la section 2.1, nous relevions qu'on définit un processus métier comme un ensemble de procédures ou d'activités qui, dans le contexte d'une organisation, réalisent un objectif particulier. La réalisation d'un ou plusieurs objectifs concrets (soins, traitement, ...) ou plus abstraits (diagnostic, pronostic, ...) est la véritable finalité du processus. Il est logique de vouloir inclure une description de ces objectifs dans le modèle de workflow qui décrit le processus.

Le besoin de flexibilité est particulièrement ressenti lorsqu'on modélise et qu'on orchestre des processus médicaux. Cette flexibilité correspond à la possibilité pour le processus d'être reconfiguré en fonction des circonstances. Nous avons vu à la section 3.9 que les techniques de modélisation basées sur des règles offraient potentiellement un niveau de flexibilité supérieure aux approches basées sur des graphes. Nous avons également constaté à la section 3.6 que les systèmes orientés-agents utilisaient une représentation interne de leurs objectifs pour être proactifs et flexibles.

6.1 Bref historique de l'approche orientée-objectifs

Plusieurs domaines reconnaissent l'importance de la représentation formelle des objectifs :

- la robotique
- le planning intelligent
- la programmation orientée agent (voir section 3.6)
- la modélisation d'organisations
- l'ingénierie des spécifications

Les approches orientées-objectifs dans le domaine de l'ingénierie logicielle et des spécifications ne sont pas neuves [96]. Depuis les années septante, il existe des méthodologies de développement qui prennent en compte les objectifs des utilisateurs et de l'organisation dans laquelle le système sera déployé ("context analysis", "definition study", "participative analysis"). A cette époque, certains

standards de documentation d'exigences comportent déjà une section consacrée aux objectifs. Dès les années 80, les objectifs sont également utilisés pour la définition de métriques ("Goal-Questions-Metrics").

Dans les années 90, les objectifs sont peu représentés dans la littérature consacrée à la spécification des logiciels et totalement absents des standards émergeants tel UML. C'est néanmoins pendant ces mêmes années que se développent des méthodologies d'ingénierie des spécifications dont l'abstraction principale est l'objectif. Ces méthodologies permettent de capturer et d'analyser les intentions des différents intervenants dans la conception d'un système d'information et d'en dériver des exigences fonctionnelles et non-fonctionnelles. Deux frameworks se distinguent particulièrement dans ce domaine :

- KAOS [22] : axé sur le raisonnement semi-formel et formel qui permet de raffiner les objectifs, les lier à des opérations, d'analyser les risques et de détecter les conflits.
- NFR/I* [57] : axé sur le raisonnement qualitatif et sur l'analyse des objectifs souples (soft goals).

Ces deux frameworks sont construits autour d'une analyse basée sur le raffinement d'objectifs en sous-objectifs. Cette décomposition de problèmes en sous-problèmes organisés selon un arbre et/ou date également des années 70. Elle trouve son origine dans l'intelligence artificielle et plus particulièrement la résolution de problèmes. KAOS (Knowledge Acquisition in Automated Specification) est sans doute le framework théorique qui pousse le plus loin la formalisation des objectifs.

Le lecteur intéressé par un historique plus complet des approches orientées-objectifs en génie logiciel peut consulter les pages 280-283 de l'ouvrage de van Lamsweerde [96].

Les objectifs ont également été introduits pour guider la modélisation d'organisations complètes [27, 41] dont les processus métiers ne constituent qu'un des composants. Dans cette optiques les objectifs correspondent aux intentions stratégiques de l'organisation. Le processus de modélisation n'est plus seulement guidé par des aspects structurels ou opérationnels mais également par la vision stratégique sous-jacente de l'organisation. Le modèle d'objectifs joue un rôle d'aide à la décision face aux options possibles de représentation. Le lecteur intéressé par la problématique de la modélisation d'organisations et de leur processus métier guidée par objectifs peut consulter l'article de Koubarakis en 1999 [41] qui esquisse les étapes d'une modélisation complète et donne un bonne vue d'ensemble des travaux antérieurs dans ce domaine.

S'il existe une littérature considérable sur la modélisation de processus métier et leur représentation sous forme de workflows, peu des ces travaux accordent une place à la capture et à la formalisation des objectifs sous-jacents aux processus.

Néanmoins, des implémentations commerciales de systèmes de gestion de workflow sont récemment apparues qui font un usage étendu des objectifs dans leurs modèles :

- GO-BPMN par Whitestein Technologies[102]
- iProcess Conductor par TIBCO [83]

L'auteur de ce travail n'a malheureusement pas eu l'occasion de les essayer pour s'en faire une opinion.

6.2 Objectifs et agents

Dans le cadre de ce travail, les définitions inspirées du framework KAOS ont été adaptées à la problématique de la modélisation d'un processus métier.

Un *agent* est un acteur du processus métier modélisé qui joue un rôle spécifique dans la réalisation des objectifs de ce processus. Les agents peuvent être des humains, des composants logiciels ou matériels. Par exemple : un infirmier, un dossier médical électronique ou un tensiomètre.

Un *objectif* est une déclaration prescriptive que le processus métier modélisé doit satisfaire à travers la coopération des agents qui le composent. Un objectif est purement déclaratif. Les objectifs doivent être exprimés en terme de phénomènes accessibles et partageables par les agents du processus. Certains agents contrôlent ces phénomènes, d'autres les surveillent.

> Par exemple, la déclaration suivante est un objectif : "La tension artérielle du patient doit être vérifiée toutes les 15 minutes".

La tension artérielle est un phénomène accessible et partageable car elle peut-être mesurée par certains agents et communiquée à d'autres. La prescription d'un médicament hypotenseur par un agent médecin permet de la faire baisser, donc d'exercer sur elle un contrôle.

On peut énoncer des objectifs à différents niveaux d'abstraction. Au plus haut niveau, ils correspondent à des objectifs stratégiques du processus métier. Par exemple :

> "Le patient doit être pris en charge"

Au plus bas niveau, les objectifs sont de nature plus technique, proches des activités du processus :

> "L'hémoglobine glycosilée du patient ne doit pas dépasser 7%"

Cette différence de niveaux d'abstraction suggère l'existence de liens de contribution entre les objectifs. Les objectifs du plus haut niveau peuvent-être raffinés en objectifs de niveau plus bas. Les objectifs du plus bas niveau peuvent être abstraits en objectifs de niveau plus élevé (voir figure 6.1).

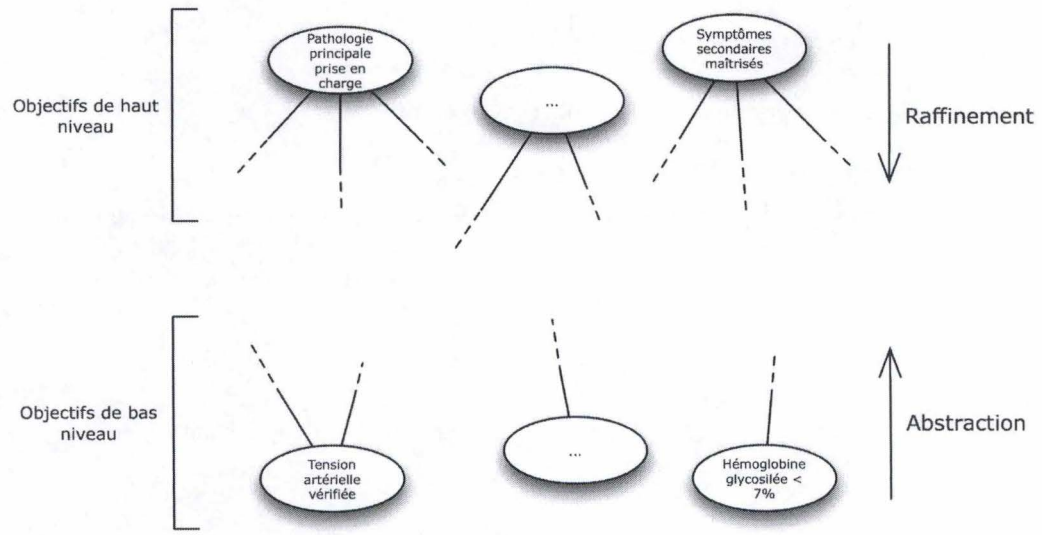

FIG. 6.1 – Abstraction et raffinement d'objectifs

Plus un objectif est de bas niveau, moins nombreux sont les agents nécessaires à sa réalisation. La prise en charge de la pathologie principale d'un patient nécessite la mobilisation d'une équipe multi-disciplinaire complète. En revanche, la vérification de la tension d'un patient ne nécessite qu'un seul soignant ou mieux, un système automatique de type Holter tensionnel.

Une *exigence* (requirement) est un objectif dont la réalisation est sous la responsabilité d'un seul agent du processus métier. "La tension du patient doit être vérifiée toutes les 5 minutes" est typiquement une exigence placée sous la responsabilité du holter tensionnel. Dans ce travail, nous nous intéressons à des workflows destinés essentiellement à des acteurs humains. Une exigence est toujours sous la responsabilité d'un acteur humain. Dans cet exemple, il s'agit de l'infirmier qui a placé le holter.

Une *attente* (expectation) est un objectif dont la réalisation est sous la responsabilité d'un seul agent qui ne fait pas partie des agents du processus métier modélisé. Le processus n'a pas de contrôle sur une attente. Dans le cadre d'un workflow hospitalier, l'objectif suivant est une attente : "La glycémie est vérifiée tous les jours à domicile" puisque cette vérification s'effectue en dehors de l'hôpital, elle échappe donc au contrôle du processus.

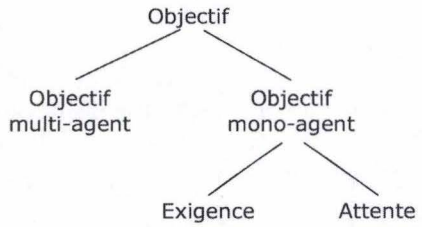

FIG. 6.2 – Objectifs, exigences et attentes (adapté de van Lamsweerde 2009 [96])

6.3 Typologie des objectifs

KAOS envisage deux types principaux d'objectifs (figure 6.3) :

- les *objectifs comportementaux* (behavioral goals)
- les *objectifs souples* (soft goals) qui prescrivent des préférences parmi des comportements alternatifs du processus métier modélisé.

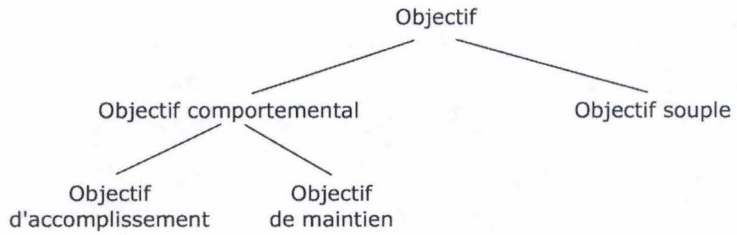

FIG. 6.3 – Typologie des objectifs en KAOS (adapté de van Lamsweerde 2009 [96])

6.3.1 Les objectifs comportementaux

Les objectifs comportementaux prescrivent déclarativement le déroulement du processus métier modélisé.

KAOS fait la distinction entre deux sous-type fondamentaux d'objectifs comportementaux : les objectifs d'accomplissement (achieve goals) et les objectifs de maintien (maintain goals). Un objectif "achieve" prescrit des comportements qui garantissent qu'un état cible sera atteint tôt ou tard moyennant que certaines conditions soient valables dans l'état courant du processus :

> Achieve[*EtatCible*] ⇔ **si** *EtatCourant* **alors tôt ou tard** *EtatCible*

Par exemple :

> Achieve[DouleurPriseEnCharge] ⇔ **si** le patient ressent de la douleur **alors tôt ou tard** cette douleur est prise en charge

Un objectif "maintain" prescrit des comportements qui garantissent qu'un état souhaité sera toujours maintenu étant donné un certain état courant du processus :

> Maintain[*EtatSouhaité*] ⇔ **si** *EtatCourant* **alors toujours** *EtatSouhaité*

Par exemple :

> Maintain[HémoglobineGlycosiléeBasse] ⇔ **si** [PatientDiabétique] **alors toujours** [HémoglobineGlycosiléeBasse]

Les objectifs comportementaux d'évitement (avoid goals) sont une variante de ce motif :

> Avoid[*EtatAEviter*] ⇔ **si** *EtatCourant* **alors toujours not** *EtatAEviter*

Dans ce travail, nous ne ferons pas la distinction entre des objectifs "achieve" et "maintain". Tous les objectifs que nous décrivons sont des entités qui peuvent être satisfaites ou pas et dont l'état de satisfaction est susceptible de changer en fonction de l'évolution du processus.

6.4 Les objectifs souples

Les objectifs souples permettent d'exprimer des préférences parmi des comportements alternatifs du processus métier. Certaines de ces alternatives les réalisent plus que d'autres. Contrairement aux objectifs comportementaux, il est difficile d'établir de façon absolue la satisfaction d'un objectif souple.

KAOS utilise des préfixes particuliers pour identifier les objectifs souples :

- Improve[EtatCible]
- Increase[QuantitéCible]
- Reduce[QuantitéCible]

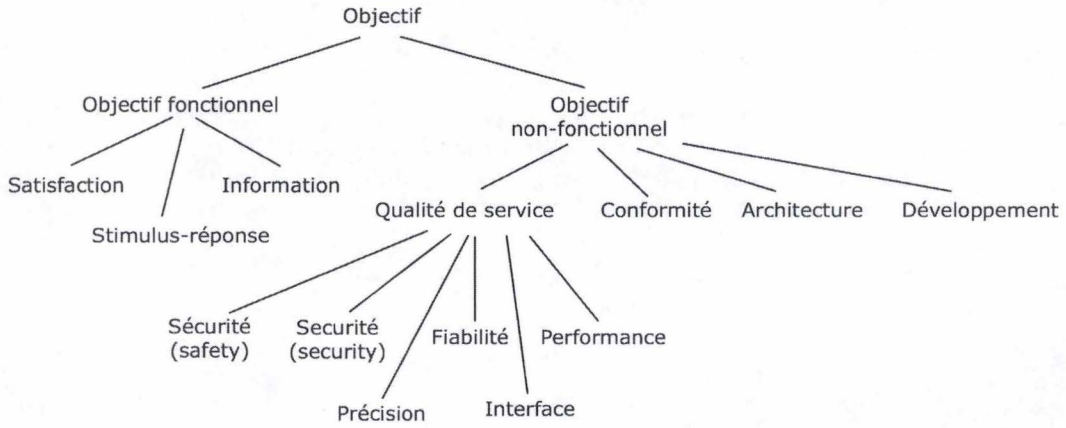

FIG. 6.4 – Principales catégories d'objectifs dans le domaine de l'ingénierie logicielle (adapté de van Lamsweerde 2009 [96])

- Maximize[FonctionObjectif]
- Minimize[FonctionObjectif]

Par exemple : Increase[FluxDePatient], Minimize[SymptômeDouloureux]

La spécification d'un objectif souple doit comporter un critère quantifiable pour permettre sa mesure. Dans l'exemple précédent "Minimize[SymptômeDouloureux]", la prescription de cet objectif doit encourager le concepteur du processus à faire usage d'une mesure quantitative de la douleur. Ce type de mesure quantitative ou semi-quantitative existe pour de nombreux symptômes.

Dans le monde des processus médicaux et plus particulièrement des itinéraires cliniques, les objectifs souples et leur évaluation correspondent aux objectifs du processus mesurés en terme d'indicateurs de qualité (voir section 4.3). Dès lors, il est intéressant de les ajouter aux modèles pour documenter explicitement les étapes du processus qui contribue à atteindre ces objectifs et durant lesquel il peut être utile d'en mesurer les indices. Du point de vue de l'auteur, ils jouent juste un rôle important de documentation.

6.5 Catégorisation des objectifs

Les catégories d'objectifs établissent une distinction entre les objectifs fonctionnels et les objectifs non-fonctionnels ainsi que leurs sous-catégories (figure 6.4). Un objectif fonctionnel fait référence aux services fournis par le processus métier. Un objectif non-fonctionnel fait référence à une qualité ou à une contrainte du processus.

Des efforts de catégorisation des objectifs ont également été menés dans le domaine médical (table 6.1). Ces catégories ne comportent que des objectifs fonctionnels qui trouvent facilement leur place dans les trois catégories fonctionnelles qui existent en ingénierie logicielle. Sans surprise, les deux sous catégories d'objectifs (connaissance et action) correspondent aux principales activités de soins : la prise de décision et la fourniture des soins proprement dit (voir section 4.3).

- Objectif de connaissance
  - Acquisition (ex. : élucider l'histoire clinique du patient, mesurer des paramètres biologiques)
  - Décision
    - Détection (ex. : surveiller l'occurrence d'un événement anormal)
    - Classification
      - Stadification (ex : déterminer le stade TNM d'un cancer)
      - Éligibilité (ex. : déterminer les critères d'accès à une étude clinique)
      - Estimation (ex. : estimer un risque de complication)
    - Prédiction
      - Diagnostic (ex. : élucider l'étiologie d'une pathologie)
      - Pronostic (ex. : évaluer la survie d'un patient)
- Objectif d'action
  - Réalisation
    - Création (ex : création d'un champ stérile)
    - Éradication (ex. : éradication d'un agent pathogène)
  - Contrôle
    - Prévention (ex. : prévention d'une complication)
    - Limitation (ex. : maintien d'un paramètre dans des limites physiologiques)
  - Communication
    - Information (ex. : communiquer des résultat)
    - Demande (ex. : demander un examen complémentaire)

TAB. 6.1 – Catégorisation des objectifs médicaux dans le domaine particulier du traitement du cancer (adapté de J. Fox 2006 [30])

Cette catégorisation gagnerait à intégrer également les objectifs non-fonctionnels liés à la qualité de service et à la conformité, deux caractéristiques importantes de processus de soins :

- Sécurité (safety) du patient (toxicité et effets secondaires des traitements, complications, infections nosocomiales, erreurs médicales, ...) et des soignants (contamination, exposition à des substances toxiques ou à des rayonnement ionisants, blessures lors des soins, burn-out, ...)
- Sécurité (security) : objectifs liés à la problématique de la confidentialité des données médicales
- Fiabilité : des machines de traitement et de diagnostic, de la nature et du conditionnement des médicaments, des informations, ...
- Performance : coût des soins, consommation des ressources et consommables divers, flux de patient, durée de séjour, files d'attentes, délais, ...
- Interface : contact avec le patient et ses proches, utilisabilité des systèmes d'information, de diagnostic et de traitement, interopérabilité entre ces systèmes, ...
- Précision : du résultat des examens cliniques et complémentaires
- Conformité : aux protocoles de soins et de traitement, aux procédures, aux guidelines, ...

Ces catégories sont utiles car elles permettent d'améliorer le processus d'élicitation des objectifs.

La revue systématique de toutes les catégories permet de découvrir des objectifs manquants. Par exemple, presque tout processus médical comporte des objectifs d'exactitude et de précision (accuracy goal). C'est d'autant plus le cas si le processus fait référence à des analyses de laboratoire ou à d'autres moyens diagnostiques techniques. Ces analyses doivent refléter le plus fidèlement possible l'état du

patient à un certain moment. Le processus doit également garantir la validité de ces données dans le temps. Par exemple, dans le cadre d'une chimiothérapie anti-cancéreuse, le taux de plaquettes du patient est très fluctuant. Un objectif d'accuracy inciterait à prévoir dans le processus des prises de sang régulières pour contrôler ce taux.

Il existe des interactions négatives entre certaines catégories. Cette information peut être exploitée pour *détecter des conflits entre objectifs*. Par exemple, dans le monde de la santé, le secret médical occupe une place importante. Les objectifs de sécurité (security goals) y sont donc fréquents. Néanmoins, la circulation de l'information au sein de l'équipe multi-disciplinaire est également primordiale pour permettre une prise en charge de bonne qualité. Dans cette situation, des conflits entre objectifs opposés sont à craindre et doivent être résolus.

Les catégories peuvent également aider à résoudre les conflits en induisant certaines heuristiques du genre : "Les objectifs de sécurité (safety goals) ont toujours la priorité lors d'une résolution de conflit". C'est d'ailleurs ce qui se passe dans la réalité puisque la sécurité du patient est une des rares cause de dérogation au secret médical.

Les catégories induisent également d'autres type d'heuristique utiles pour *découvrir des objectifs supplémentaires*. Par exemple, les objectifs de sécurités (safety) sont en général des objectifs d'évitement (avoid goal) de complications, d'infections ou d'erreurs. Ils peuvent également correspondre à des objectifs "achieve" liés à toutes les activités de préventions des risques.

Les types et les catégories d'objectifs sont des classifications indépendantes. Il existe néanmoins certaines relations entre elles :

- les objectifs fonctionnels sont souvent des objectifs "achieve".
- les objectifs de sécurité ("safety" et "security") et de précision ("accuracy") sont souvent des objectifs "maintain".
- les objectifs liés à des notions de coût, de consommation de ressources ou aux performances en général sont souvent de objectifs souples.

6.6 Découverte d'objectifs dans les processus de soins

Nous avons dit que les processus médicaux sont complexes. La figure 6.5 donne une vision schématique du processus de prise en charge d'un patient atteint d'un cancer colo-rectal. Ce processus est un itinéraire clinique réellement mis en place dans un grand hôpital bruxellois et à la formalisation duquel l'auteur de ce travail a participé. Chacune des boîtes est une activité. La plupart de ces activités ne sont pas atomiques et peuvent être décomposées en sous-activités. Dans cet exemple, il est possible de descendre jusqu'à 4 niveaux de raffinement en dessous de celui présenté. Dans la réalité, ce processus comporte plusieurs centaines d'activités. L'activité mise en évidence correspond à l'étape de traitement chirurgical du patient. En "ouvrant la boîte", on visualise les sous-activités qui la composent.

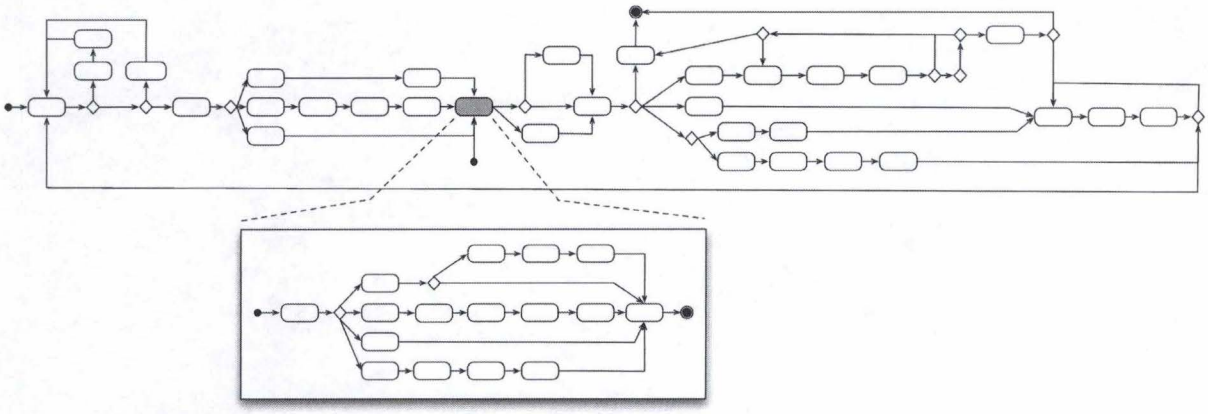

FIG. 6.5 – Vision schématique d'un itinéraire clinique et raffinement d'une de ses activités

Cet exemple gagnerait en clarté si on le modularisait un peu plus. Néanmoins, la vision complète de ce premier niveau permet d'embrasser d'un seul tenant, les grandes étapes de la prise en charge du patient (figure 6.6) : le diagnostic, le traitement et le suivi. Ces étapes peuvent elles-mêmes être raffinées en sous-étapes. Par exemple, le traitement du patient consiste tout d'abord à diminuer la taille de la tumeur (radiothérapie et chimiothérapie adjuvante), retirer la tumeur (chirurgie) et éliminer d'éventuelles cellules cancéreuses résiduelles (radiothérapie et chimiothérapie post-chirurgicales) (figure).

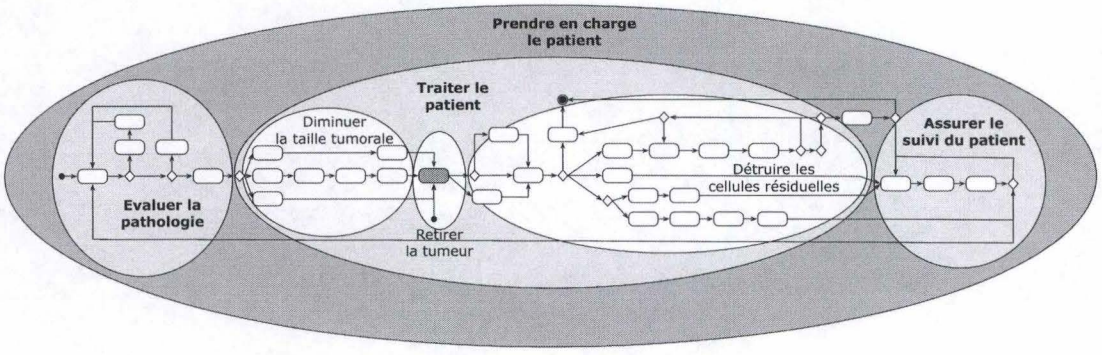

FIG. 6.6 – Grandes étapes fonctionnelles d'un itinéraire clinique de prise en charge d'un cancer

Chacune de ces étapes concoure à la réalisation d'un objectif particulier. Dans la terminologie KAOS, il s'agit d'objectifs comportementaux de type "achieve" (figure 6.7) : Achieve[PatientPrisEnCharge], Achieve[PathologieEvaluée], Achieve[PatientTraité], ...

Comme suggéré à la section 6.2, il existe des relations de contribution entre ces objectifs. Ces relations induisent une hiérarchie de raffinement dans laquelle des objectifs fils contribuent à la réalisation de leurs objectifs pères (figure 6.8).

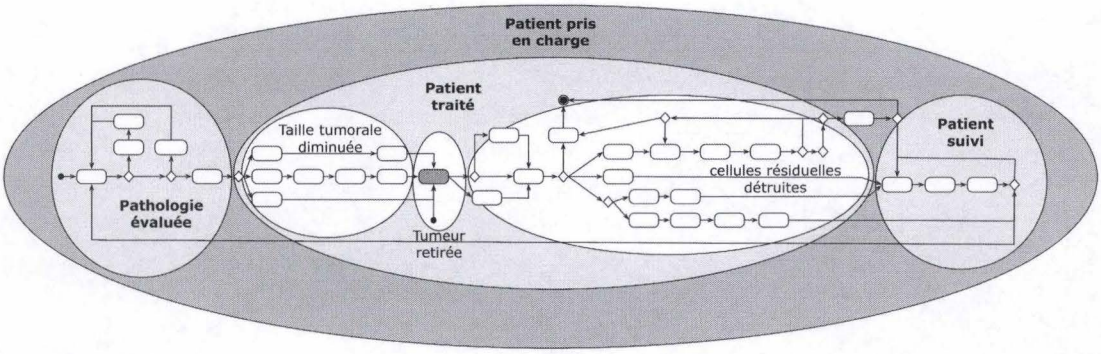

FIG. 6.7 – Principaux objectifs comportementaux d'un itinéraire clinique de prise en charge d'un cancer.

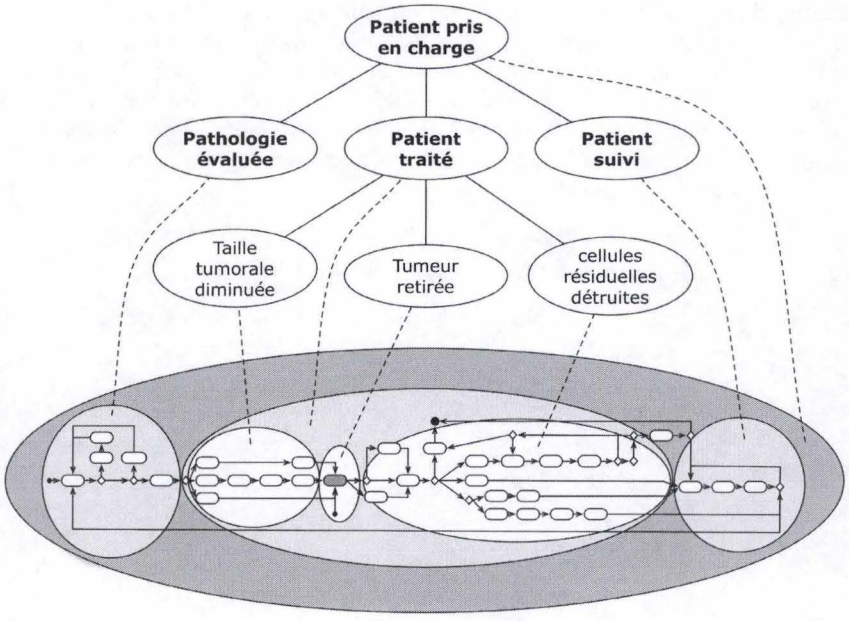

FIG. 6.8 – Hiérarchie de raffinement des objectifs comportementaux de haut niveau d'un itinéraire clinique

Naturellement, il est tentant de raccrocher aux objectifs inférieurs de la hiérarchie, les fragments de workflow qui concourent à les réaliser (figure 6.9). A partir de maintenant, nous appellerons "procédures" ces fragments de workflow.

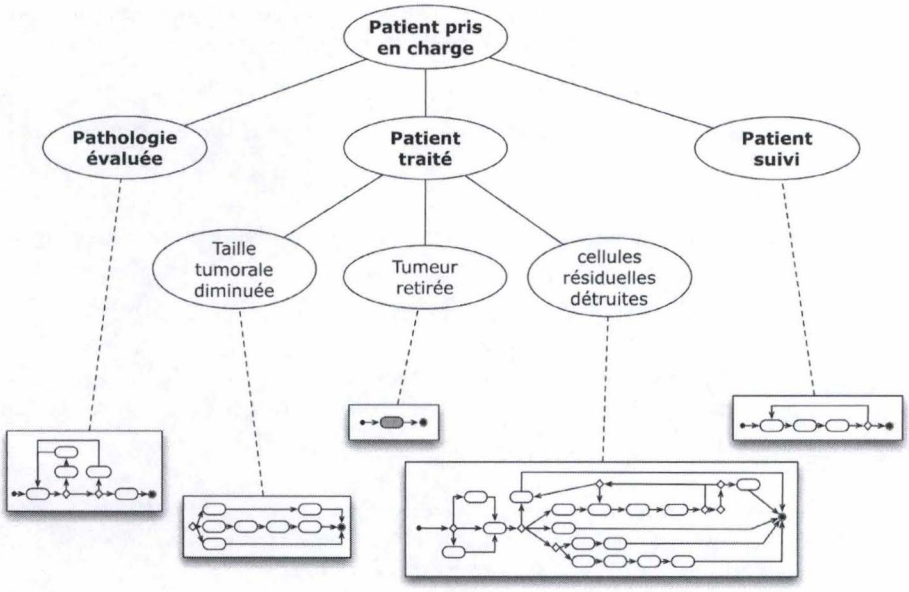

FIG. 6.9 – Processus erroné de découpage d'un workflow en procédures indépendantes

Ce processus de fragmentation ou de modularisation du workflow en procédures n'est pas aussi simple. On constate que certaines transitions entre activités ont été rompues. Par exemple, la transition qui existait entre le noeud de décision situé tout à droite du modèle et la première activité tout à gauche n'existe plus. Dans l'itinéraire clinique réel, cette transition est très importante car elle indique qu'une récurrence du cancer diagnostiquée pendant la phase de suivi doit refaire l'objet d'un diagnostic et d'un traitement complet. Dans un workflow réel, il existe des dépendances liées au flux de contrôle entre des activités parfois très éloignées "géographiquement" dans le modèle. Cette observation suggère que les procédures doivent être conçues spécifiquement pour pouvoir se raccrocher à la hiérarchie d'objectifs.

6.7 Séances d'ingénierie de processus

A la section précédente, nous avons déconstruit un processus de soins complet en procédures à partir des grands objectifs ou étapes fonctionnelles de ce processus. Dans la réalité, les choses ne se passent habituellement pas de cette façon. Même si un processus est déjà en fonctionnement dans une organisation de soins, il n'y est presque jamais formalisé. Une option possible et qui se révèle fructueuse quand elle est mise en pratique, consiste à élaborer un modèle d'objectifs en parallèle de la formalisation du workflow.

La tâche de l'analyste consiste à élucider et formaliser la façon dont les gens travaillent sur le terrain. Ce travail d'enquête est émaillé d'interviews individuelles ou collectives, par petits groupes ou parfois avec l'ensemble de l'équipe multidisciplinaire pour reconstituer le processus suivi.

Selon l'expérience de l'auteur, deux types d'éléments ressortent de séances de travail individuel avec les acteurs du processus de soins :

- l'agent explique la façon dont il travaille, les activités qu'il effectue sur le terrain. Le résultat de ces interviews sont des fragments de procédures vus à travers la perspective unique de l'agent interrogé.

- des objectifs de bas niveau souvent exprimés sous la forme : "j'effectue ces activités pour atteindre ce résultat".

Les séances de travail en groupe ramènent plutôt des informations intentionnelles et opérationnelles de haut niveau :

> "l'itinéraire clinique doit offrir la meilleure prise en charge de la pathologie principale du patient tout maîtrisant les effets secondaires des traitements et en lui garantissant une qualité de vie acceptable."

> "L'itinéraire clinique de traitement du cancer du sein comporte cinq étapes principales : la découverte et le diagnostic de la pathologie, le traitement pré-chirurgical, la chirurgie d'exérèse de la tumeur, le traitement post-chirurgical et le suivi."

Sans généraliser, la nature de ces informations dépend également du rôle endossé par l'agent interviewé :

- Les médecins ont tendance à donner des objectifs et des plans de haut niveau, même s'ils peuvent être d'extrêmement bas niveau dans leur domaine de compétence particulier.

- Les infirmiers et le personnel paramédical (kinésithérapeute, psychologue, techniciens, ...) ont plutôt tendance à décrire des objectifs et des plans de bas niveau.

C'est à l'analyste qu'il incombe de rassembler ces informations en un tout cohérent.

6.8 Graphe de raffinement ET/OU

Nous avons vu que les objectifs d'un processus trouvent naturellement leur place dans une hiérarchie de relations de contribution. On peut représenter plus formellement cette hiérarchie sous la forme d'un graphe de raffinement ET/OU. Les objectifs constituent les noeuds du graphe. Deux types de relations fondamentales existent entre ces noeuds : Des relations de raffinement ET et des relations de raffinement OU (figure 6.10). Ces relations lient un objectif à un ensemble de sous-objectifs. Dans la relation, chaque sous objectif contribue à la réalisation de son objectif père.

Une relation de raffinement ET signifie que l'ensemble des sous-objectifs doit être satisfait pour considérer que l'objectif père est lui-même satisfait. Une relation de raffinement OU signifie qu'il suffit qu'un seul des sous-objectifs soit satisfait pour considérer que l'objectif père est satisfait.

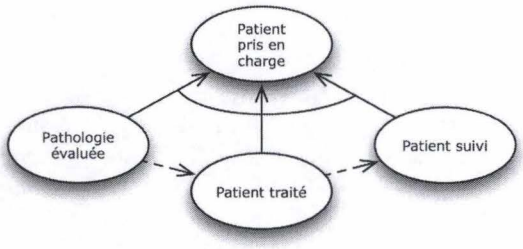

FIG. 6.11 – Relations de précédences entre objectifs

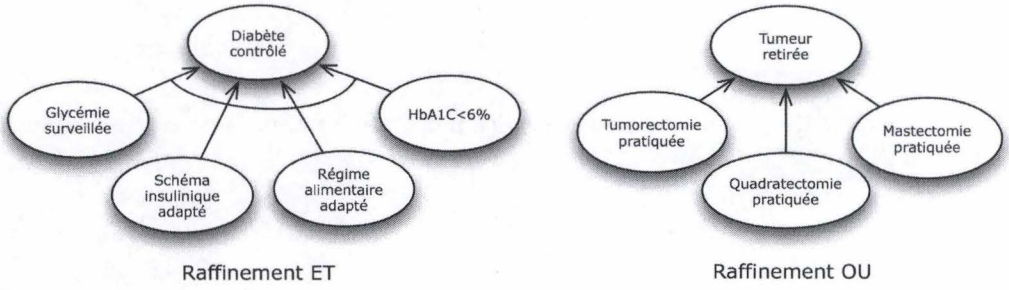

FIG. 6.10 – Relations de raffinement ET et OU

D'autres types de relations peuvent être rajoutés au graphe de raffinement :

- des relations de précédence : un objectif doit être temporellement satisfait avant un autre. Par exemple, un objectif lié à la collecte de données doit être considéré avant un objectif lié à la prise d'une décision.
- des relations de priorité : un objectif doit être satisfait à la place d'un autre. Par exemple, traiter efficacement le patient est plus important que d'assurer son confort.

Nous n'aborderons pas la problématique des relations de priorités entre objectifs dans ce texte. Par contre, des relations de précédences sont indispensables si l'on veut utiliser le modèle d'objectif pour contraindre l'exécution du workflow. Une façon de représenter ces relations de précédence est illustrée à la figure 6.11. Les flèches pointillées indiquent que le noeud à l'origine de la flèche doit être atteint avant le noeud à l'extrémité.

Le graphe de raffinement est un graphe dirigé acyclique. C'est un graphe car il peut y avoir plusieurs objectifs racines de la hiérarchie et un objectif peut avoir plusieurs pères auquels il contribue. Acyclique parce qu'un objectif ne peut pas contribuer à un autre situé plus bas dans sa hiérarchie de raffinement.

Cette approche de raffinement des objectifs dans un graphe est à rapprocher d'un paradigme standard de résolution de problème en intelligence artificielle par décompositions ET-OU successives.

6.8.1 Heuristique de raffinement

Il existe une heuristique simple mais qui permet de construire systématiquement l'arbre de raffinement. Nous avons vu que les interviews avec les acteurs de terrain ramènent des objectifs de niveaux variables. Au sortir de ces interviews, le modèle d'objectifs ressemble à un nuage d'objectifs épars ou vaguement connectés. Chacun de ces objectifs doit faire l'objet d'une analyse particulière.

On identifie les pères d'un objectif en posant la question "pourquoi" ?

> Pourquoi la tumeur doit-elle être visualisée ?
>
> Réponses : Pour la localiser dans le sein, pour mesurer son grand axe, pour évaluer sa forme (spiculaire ou nodulaire), pour détecter la présence de microcalcifications, ...

La figure 6.12 présente le fragment de modèle d'objectif qui résulte de cette première question.

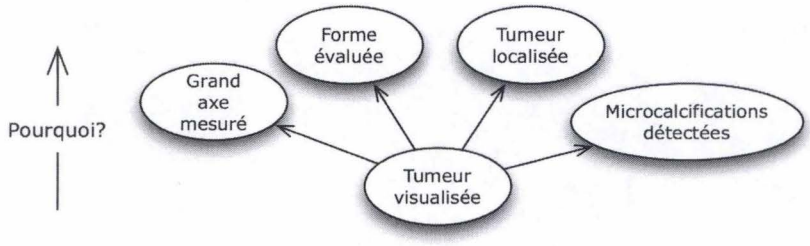

FIG. 6.12 – Découverte des "pères" d'un objectif

On identifie les fils d'un objectif en posant la question "comment ?" :

> Comment visualiser la tumeur ?
>
> Réponses : en pratiquent une échographie, une mammographie, une résonance magnétique nucléaire, ...

La figure 6.13 présente le fragment de modèle d'objectifs qui résulte de cette seconde question.

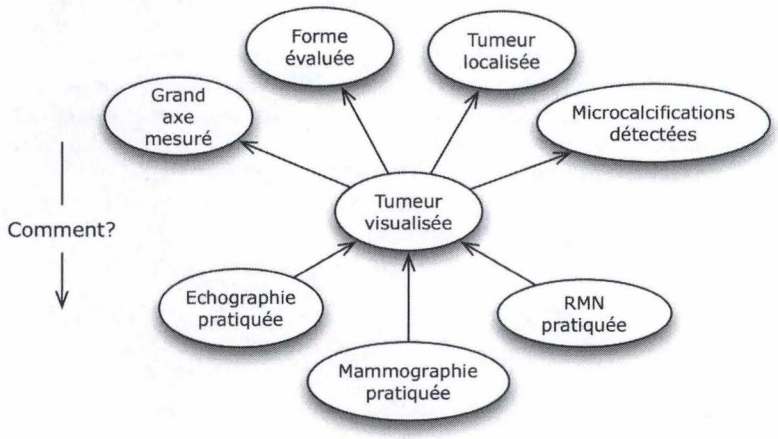

FIG. 6.13 – Découverte des "fils" d'un objectif

Pour chacun de ces sous-objectifs détectés, on peut ensuite se demander si cet objectif est suffisant pour satisfaire l'objectif père ou si d'autres sont également nécessaires. Dans cet exemple, chaque objectif est suffisant. Il s'agit d'un raffinement OU.

La même analyse est pratiquée pour chacun des nouveaux objectifs identifiés. Le graphe de raffinement se construit ainsi peu à peu. A la fin de cette analyse, un balayage systématique des catégories d'objectifs non-fonctionnels (voir section 6.5) ramène encore des objectifs supplémentaires qui doivent trouver leur place dans la hiérarchie. Ce processus de construction est bien évidemment itératif et pour simple qu'il semble être, demande une certaine pratique pour obtenir des modèles de qualité.

Nous avons évoqué le fait que les interviews ramenaient également des fragments de procédures de différents niveaux. Au fil du processus d'ingénierie du processus, les étapes des procédures de haut niveau deviennent en général des objectifs fonctionnels ou sont raffinées en sous-étapes de plus bas niveaux. Ces étapes de plus bas niveau iront s'intégrer aux procédures qui réalisent les objectifs feuilles du graphe.

Cette courte section n'a fait que survoler très brièvement les principes généraux de construction du graphe raffinement. Pour une exploration plus approfondie de ces techniques, nous invitons le lecteur intéressé à consulter le chapitre 8 de l'ouvrage de van Lamsweerde [96].

6.9 Critère de terminaison du raffinement

Une question que l'analyste ne manquera invariablement pas de se poser est de savoir quand ce processus de raffinement des objectifs doit prendre fin ?

A l'inspiration de KAOS, l'auteur propose de suivre l'heuristique suivante : un objectif ne doit plus être raffiné quand il peut-être réalisé par au moins une procédure bien identifiée et dont la bonne exécution est placée sous la responsabilité d'un agent ou un rôle unique lui-même bien identifié. Cela n'implique pas qu'il n'y ait qu'un seul rôle qui exécute réellement la procédure. D'autres rôles ou agents adjuvants sont très souvent requis. Néanmoins, c'est ce seul rôle ou agent qui est responsable du résultat.

La figure 6.14 est illustrative de ce critère. A chaque objectif feuille de ce fragment de modèle est associé au moins une procédure qui le réalise. Chacune de ces procédures est placée sous la responsabilité d'un seul rôle responsable. Une procédure particulière comporte une séquence coordonnée d'activités qui ne sont pas détaillées dans le modèle mais peuvent faire l'objet d'un workflow particulier (figure 6.15). Par exemple, la procédure de consultation en oncologie consistera en un passage à l'accueil de la clinique pour faire éventuellement renouveler sa carte de l'hôpital, un passage à l'accueil d'oncologie pour remplir certaines formalités, ... Plusieurs agents adjuvants participent à cette procédure (accueillante, secrétaire, ...) mais au final, c'est l'oncologue qui est responsable de la tenue de cette rencontre avec le patient.

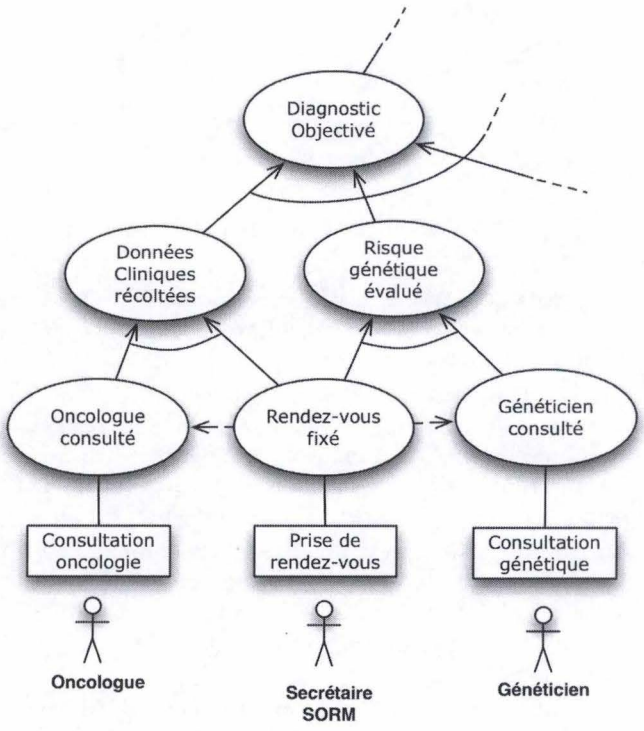

FIG. 6.14 – Critère de terminaison du raffinement : objectif réalisé par au moins une procédure dont l'exécution est placée sous la responsabilité d'un agent unique.

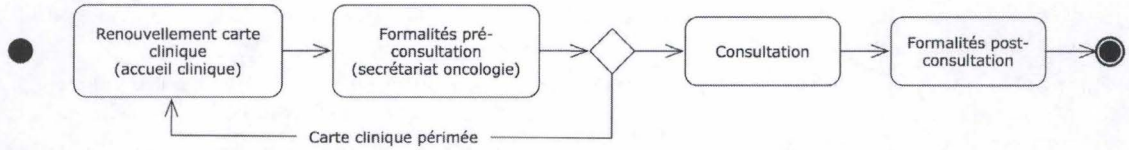

FIG. 6.15 – Workflow de la consultation en oncologie

Selon l'expérience de l'auteur, l'adoption de ce critère de terminaison facilite la formalisation de la procédure particulière qui réalise l'objectif. Les agents qui endossent le rôle de responsable ont en général une bonne connaissance de la procédure et des agents adjuvants impliqués. De plus, ne devoir dialoguer qu'avec quelques responsables bien identifiés facilite les choses.

Un fois l'agent (ou les agents) responsable identifié, on peut l'interviewer pour déterminer avec lui la succession coordonnée des tâches qui composent la procédure. Si l'agent responsable n'a pas une connaissance suffisante de certains détails opérationnels, on peut également interviewer les agents adjuvants identifiés par l'agent responsable.

Ainsi, le processus d'élucidation et de formalisation de la séquence coordonnée des activités de la procédure peut se résumer de la façon suivante :

1. interview de l'agent responsable et élaboration d'une première ébauche de modèle

2. interview de chacun des agents adjuvants et amélioration du modèle

3. séance de validation du modèle par l'équipe entière (agent responsable + agents adjuvants), correction et validation finale.

Si la procédure est réellement complexe, cette séquence peut être réalisée itérativement jusqu'à ce qu'on obtienne un modèle validé par tous. Dans le cas de procédures simples, on peut bien entendu brûler des étapes. A toute règle empirique comme celle-ci, il y a bien entendu des exceptions.

6.10 Réalisation des objectifs par des procédures

Le critère de terminaison évoqué à la section précédente implique qu'au final, tout objectif feuille du graphe est réalisé par au moins une procédure. Si l'objectif feuille est une exigence, la procédure identifiée fait partie du workflow du processus. Si l'objectif feuille est une attente, la procédure qui le réalise ne s'exécute pas au sein du processus. Néanmoins, il est intéressant de la représenter car cette procédure constitue un point de contact entre le processus et le monde extérieur. Elle peut éventuellement correspondre à une interface entre le système de gestion de workflow et une application externe (figure 2.5). On représente une attente par un ovale pointillé. La ou les procédures qui réalisent l'attente sont également représentées en pointillés (voir figure 6.16).

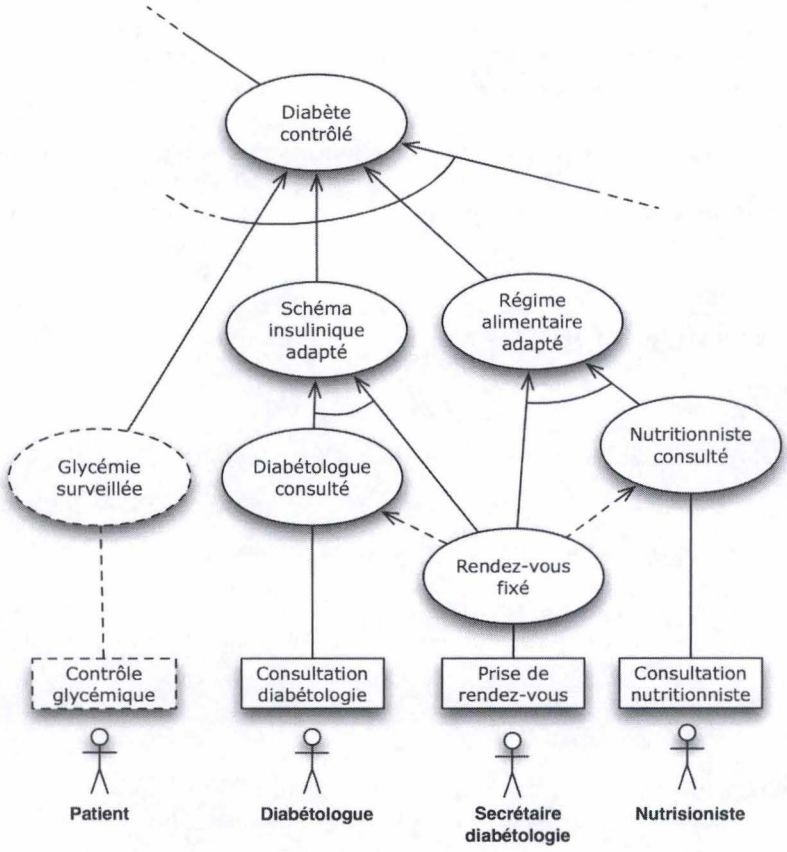

FIG. 6.16 – Représentation d'une attente et de sa procédure externe

Dans cet exemple, le patient réalise son contrôle glycémique lui-même chez lui. Cette procédure échappe donc au workflow du processus hospitalier. Comme on n'a aucun contrôle sur la satisfaction de cette attente, il est préférable de ne pas tenir compte de sa satisfaction pour la satisfaction de l'objectif père ou de considérer qu'elle est toujours satisfaite. Sa représentation sur le diagramme tient de la documentation.

Néanmoins, dans cet exemple, la surveillance de la glycémie est une composante importante du contrôle du diabète. D'un commun accord avec le diabétologue, l'analyste peut décider de modifier le workflow de l'activité de consultation diabétologique pour y inclure une étape de vérification des chiffres de glycémie dont le patient aura pris note. La procédure de consultation satisfait maintenant deux objectifs et il n'est plus nécessaire de modéliser l'intervention du patient, extérieure au workflow (figure 6.17).

Nous venons de voir qu'une même procédure peut satisfaire plusieurs objectifs. Il est également possible que plusieurs procédures satisfassent un même objectif.

Dans le fragment de modèle suivant 6.18, les deux procédures $Mammographie$ et $IRMSein$ concourent à la réalisation du même objectif de visualisation de la tumeur. Si ces deux procédures ont le même

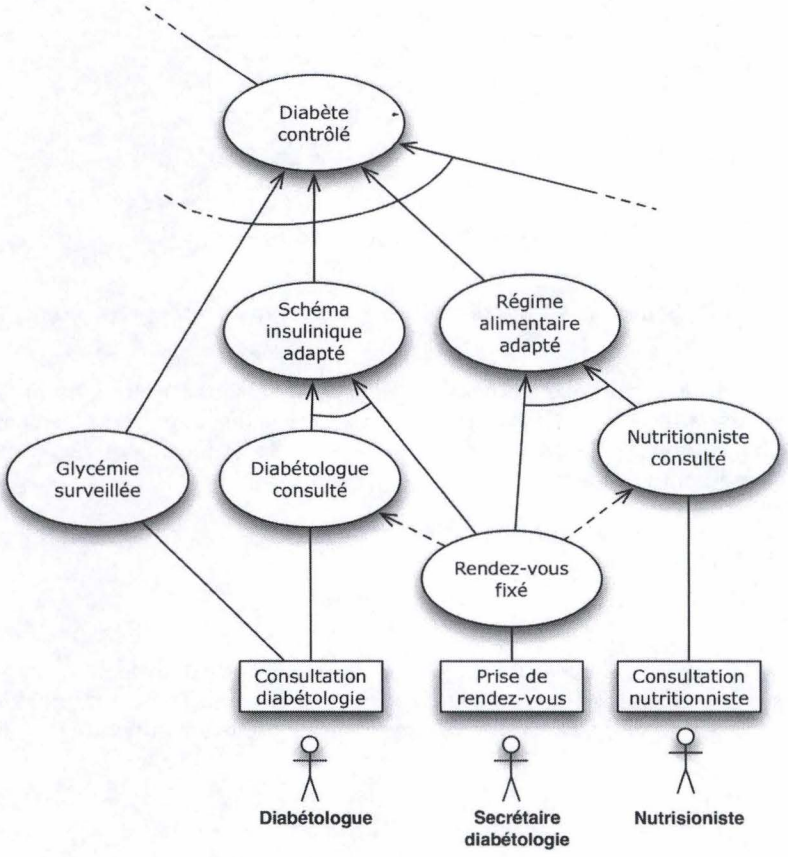

FIG. 6.17 – Modification de la procédure de consultation diabétologique pour prendre en charge le contrôle glycémique et supprimer l'attente.

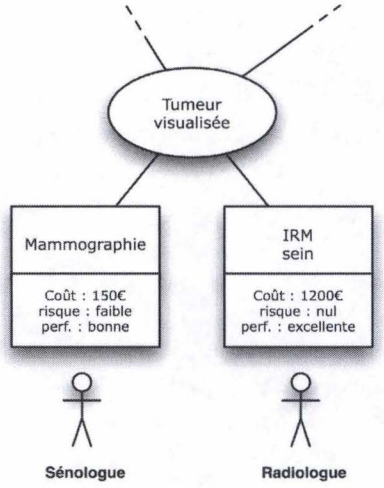

FIG. 6.18 – Métadonnées adjointes aux procédures

objectif, leur choix n'est pas indifférent. L'IRM (imagerie par résonnance magnétique nucléaire) offre une bien meilleure visualisation et a l'avantage de ne pas irradier la patiente. En revanche, la mammographie est moins coûteuse. Pour documenter ces avantages et inconvénients respectifs, l'analyste peut adjoindre des métadonnées aux procédures :

- Coût : valeur numérique estimée en euros (par exemple)
- Risque : nul, faible, moyen, élevé
- Performance : mauvaise, bonne, excellente
- ...

Dans certains cas, le choix ne se pose même pas et les conditions de sélection de la procédure font partie de ses préconditions. Dans l'exemple représenté à la figure 6.19, la tumorectomie est réservée aux cancers peu avancés. Nous pourrions donc retrouver les prédicats suivants :

$$\ldots \wedge (HoldsAt(StadificationCancer(patient, Sein, T1), t) \vee$$
$$(HoldsAt(StadificationCancer(patient, Sein, T2), t)) \wedge \ldots$$

dans les préconditions de la procédure $Tumorectomie$.

6.11 Avantages d'une approche orientée-objectifs

L'expérience de l'auteur suggère que la construction d'un modèle d'objectifs comme abstraction centrale pour modéliser des workflows facilite l'élucidation du travail des acteurs sur le terrain et la conception du modèle de processus. La technique de raffinement des objectifs dans un graphe ET-OU et leur opérationalisation sous forme de procédures fournit un mécanisme naturel de construction du workflow. Cette pratique est en phase avec le raisonnement clinique où un médecin choisi entre différentes stratégies thérapeutiques en fonction d'objectifs qu'il nourrit pour son patient.

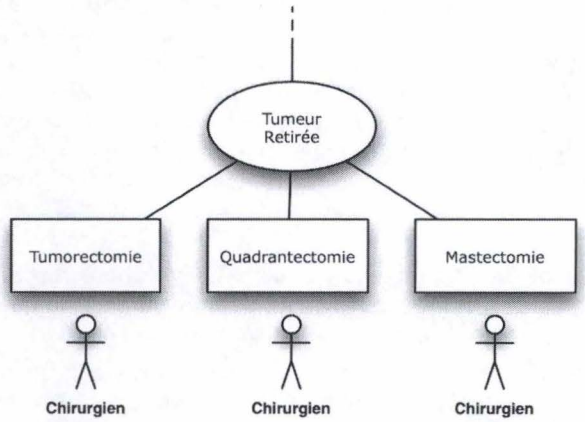

FIG. 6.19 – Procédures dont la sélection dépend de certaines préconditions

Un modèle d'objectifs permet d'exprimer les processus de manière beaucoup plus déclarative. Les détails techniques et procéduraux demeurent encapsulés dans les procédures. Le caractère modulaire des procédures permet éventuellement leur réutilisation. Ceci est par exemple vrai pour les procédures administratives, logistiques ou liées à des soins stéréotypés (prise de sang, demande d'examen complémentaire, ...) qui peuvent être réutilisées d'un processus de soins à l'autre.

On peut ajouter des procédures et des objectifs au modèle en préservant la validité globale du workflow. Ceci rend possible une évolution continue d'un modèle lorsque le processus réel est en cours d'exécution sur le terrain. Cette caractéristique est très importante pour des processus de soins qui peuvent durer des années (le suivi de certains cancers s'étend sur plus de vingt ans). Pareille flexibilité du modèle nous semble difficile à atteindre avec des workflows purement procéduraux.

Les objectifs fournissent une justification du bien fondé des procédures. Comme les objectifs font partie d'une hiérarchie, on constate facilement la contribution respective de chaque procédure à des objectifs de haut niveau. Il est d'ailleurs utile de rendre visible le modèle d'objectifs aux acteurs du processus qui exécutent les procédures sur le terrain. Ce modèle leur permet de situer leur rôle dans le projet global poursuivi pour le patient. Cette visibilité contribue à motiver les acteurs (y compris le patient comme acteur de son propre processus de soins) à effectuer certaines activités par la compréhension des enjeux sous- jacents.

La réalisation des objectifs lors de l'exécution du workflow fournit un critère précis d'avancement du workflow. Les objectifs constituent des jalons naturels à atteindre dans le déroulement d'un processus. Les paramètres liés à ces jalons constituent d'excellents indices de performance : nombre d'objectifs atteints, en combien de temps, à quel coût et en consommant quelles ressources, etc. Une corrélation entre les procédures exécutées et les objectifs atteints permet également une analyse de performance de chaque procédure individuelle. Les procédures les moins efficaces peuvent faire l'objet d'une redéfinition. Les résultats de ces analyses alimentent la démarche d'amélioration continue du processus.

Enfin, les objectifs constituent une excellente base de réfléxion à propos d'options opérationnelles à introduire ou pas dans le workflow.

6.11.1 Stabilité et transposabilité du modèle d'objectifs

L'auteur constate que le modèle d'objectifs est un modèle stable, qui pour un même processus de traitement est transposable d'une organisation de soins à l'autre. Par contre, seules certaines procédures du modèle opérationnel sont transposables. Le plus souvent, il s'agit de procédures en rapport avec des scénarios de soins qui font l'objet de consensus. Il peut s'agir par exemple de protocoles de traitement dérivés des guidelines. Le plus souvent, même ces scénarios nécessitent des adaptations locales qui tiennent compte des infrastructures, des ressources, du personnel disponible ou de l'expérience de certains acteurs.

Par exemple, la grande majorité des processus de soins hospitaliers répond aux objectifs suivants : prendre en charge un patient en lui assurant les meilleures chances de guérison, tout en lui offrant un confort acceptable et en gaspillant le moins de ressources possible.

Au vu des ressources investies dans la définition des itinéraires cliniques, leur transposition d'un hôpital à l'autre semble être un enjeu important. C'est d'ailleurs une des raisons qui justifie l'existence en Belgique et au Pays-bas du Réseau inter-hospitalier des Itinéraires Cliniques (RIC)

6.12 Représentation formelle des procédures

Nous avons vu au chapitre 5 qu'il était possible de représenter la séquence des activités coordonnées dans une procédure avec des formules du calcul des événements. Néanmoins, en l'absence d'outil qui supporterait un langage de spécification de plus haut niveau, la construction de ce type de modèle s'avère fastidieuse.

Dans ce chapitre, nous prenons le parti de dire qu'il n'est pas absolument nécessaire de décrire l'ensemble des activités des procédures pour les connecter au modèle d'objectifs. Une description de leurs pré et post-conditions suffit pour effectuer des raisonnements intéressants quant au démarrage et à la terminaison des procédures.

6.12.1 Axiomes de changement d'état d'exécution

Les axiomes suivants déterminent les deux états d'exécution d'une procédure ($EnCours$ et $Terminée$) et l'effet des événements $StartProcédure$ et $EndProcédure$ sur ces états :

$$Initiates(StartProcédure(procédure, patient), EnCours(procédure, patient), t)$$
$$Terminates(StartProcédure(procédure, patient), Terminée(procédure, patient), t)$$
$$Initiates(EndProcédure(procédure, patient), Terminée(procédure, patient), t)$$
$$Terminates(EndProcédure(procédure, patient), EnCours(procédure, patient), t)$$

Encore une fois, dans des modèles plus évolués, nous envisagerions plus de deux états d'exécution pour une procédure : procédure interrompue, procédure annulée, ... (voir chapitre 5.12).

Ces axiomes permettent à une procédure terminée de retourner néanmoins dans l'état $EnCours$. Il s'agit d'une simplification nécessaire car nous n'introduisons pas dans ce texte la notion d'identifiant d'instance de procédure. Une procédure peut donc être exécutée plusieurs fois. Cette simplification a pour effet d'empêcher plusieurs instances de la même procédure de s'exécuter en même temps. Il s'agit clairement d'une importante limitation du modèle qui ne peut être levée qu'en adoptant une représentation des procédures qui incorpore la notion d'instance.

Les remarques à propos des conventions de nommage exprimées à la section 5.12.2 ont également cours ici. De façon simplifiée, une procédure est identifiée ici par un nom de procédure (ex. *ConsultationOncologue*) et un identifiant de patient. Il conviendrait également d'ajouter un identifiant d'instance du processus dans laquelle se déroule la procédure et un identifiant d'instance de procédure. Des informations à propos des agents ou des rôles chargés d'exécuter la procédure manquent également. La gestion de ces identifiants d'instance d'exécution est délicate et sujette aux erreurs. Ces identifiants devraient être générés automatiquement, par exemple par un interpréteur de workflow chargé de l'exécution du modèle.

6.12.2 Préconditions

Les préconditions spécifient le contexte nécessaire pour qu'une procédure puisse démarrer. Il peut s'agir de conditions liées au patient ou à l'environnement du processus, par exemple la disponibilité de certaines ressources. L'exemple suivant décrit les préconditions d'une procédure *SéanceRadiothérapieSein* :

$\neg HoldsAt(EnCours(SéanceRadiothérapieSein, patient), t) \wedge$: la procédure ne peut démarrer si elle est déjà en cours.

$HoldsAt(RendezVous(patient, SéanceRadiothérapieSein, t), t) \wedge$: un rendez-vous a bien été pris pour cette période.

$HoldsAt(Libre(Clinac6EX), t) \wedge$: la machine de traitement est disponible à ce moment là.

⋮

$\wedge HoldsAt(Pathologie(patient, CancerSein), t) \Rightarrow$: cette procédure est réservée aux patientes atteintes d'un cancer du sein.

$Happens(StartProcédure(SéanceRadiothérapieSein, patient), t) \wedge$: démarrage de la procédure.

$Happens(Occupation(Clinac6EX), t) \wedge$: la machine de radiothérapie est maintenant occupée.

$Happens(EndProcédure(procédure, patient), t + 20) \wedge$: fin de la procédure 20 minutes plus tard.

La spécification d'une durée fixe d'exécution de la procédure est également une simplification. Si l'on décide d'incorporer dans le modèle le descriptif complet de la succession des activités, cette durée sera variable en fonction des chemins pris dans la procédure et de la durée des activités effectuées. Une autre possibilité consiste à considérer que la procédure a une durée comprise dans un certain intervalle de temps $t_1 \leftrightarrow t_2$ (ici >10 et < 30 minutes) :

$$\begin{aligned}
&\neg HoldsAt(EnCours(SéanceRadiothérapieSein, patient), t_1 - 1) \wedge \\
&HoldsAt(EnCours(SéanceRadiothérapieSein, patient), t_1) \wedge \\
&t_2 > t_1 + 10 \wedge \\
&t_2 < t_1 + 30 \Rightarrow \\
&Happens(EndProcédure(procédure, patient), t_2)
\end{aligned}$$

Si cette façon de représenter la fin d'une procédure est plus proche de la réalité, elle a également tendance à faire "exploser" la taille et le nombre de scénarios produits par le moteur de raisonnement. C'est la responsabilité de l'analyste de trouver un juste milieu entre la précision du modèle, l'usage qu'il veut en faire et la possibilité matérielle de réaliser les calculs et d'en exploiter les résultats.

De la même façon, l'incorporation de la description complète des procédures sous la forme de séquences coordonnées d'activités semble difficile à gérer par l'implémentation actuelle du moteur de raisonnement fournie par Mueller.

Notre exemple illustre également la représentation d'une ressource (ici une machine de radiothérapie). Dans ce texte, nous n'aborderons pas la construction d'un modèle de ressources digne de ce nom dans lequel des ressources peuvent être monopolisées transitoirement, complètement ou pas mais également être consommées. Dans notre exemple, nous représentons la ressource à l'aide de deux fluents ($Occupée(r)$, $Libre(r)$) paramétrés par un identifiant. Deux événements associés ($Libération(r)$, $Occupation(r)$) assurent les changements d'états.

6.12.3 Post-conditions et satisfaction d'objectifs

Il existe un lien entre le modèle opérationnel des procédures et le modèle intentionnel des objectifs. Ce lien est matérialisé par le fait que certaines procédures ou certaines activités contenues dans des procédures satisfont certains objectifs. Dit autrement, la satisfaction de l'objectif fait partie des post-conditions de la procédure.

L'exemple suivant explicite certaines des post-conditions de la séance de radiothérapie dont les pré-conditions sont décrites à la section précédente :

$\neg HoldsAt(Terminé(SéanceRadiothérapieSein, patient), t-1) \wedge$
$HoldsAt(Terminée(SéanceRadiothérapieSein, patient), t) \Rightarrow$
$Happens(Satisfaction(FractionDélivrée, patient, ICCancerDuSein), t) \wedge$: l'objectif $FractionDélivrée$ est atteint
$\vdots$
$\wedge Happens(Libération(Clinac6EX), t)$: et la machine est libre pour le patient suivant.

Les post-conditions des procédures peuvent donc concerner la satisfaction des objectifs du processus ou des modifications de l'état du patient et de l'environnement.

Dans ce mode de représentation, les effets d'une procédure sur la satisfaction des objectifs, l'état du patient et de l'environnement sont déterministes. C'est-à-dire que ces effets sont toujours constatés lorsque la procédure arrive à son terme. En médecine, la plupart des effets sur l'état du patient ne sont pas déterministes. C'est le cas des effets des traitements qui peuvent réussir et donc provoquer l'effet attendu, échouer et donc ne pas provoquer l'effet attendu ou provoquer des effets secondaires en plus de l'effet attendu ou pas. Il est donc, en général, assez délicat d'introduire des effets sur l'état du patient dans les post-conditions des procédures. Les effets sur l'environnement sont en général beaucoup plus déterministes et l'on peut sans crainte, inférer qu'une machine de radiothérapie redevienne libre à la fin d'une séance de traitement.

Si cette modélisation déterministe de la réalité est jugée trop simpliste, au-moins deux possibilités s'offrent à l'analyste :

- S'il consent à l'effort de représentation de toutes les activités coordonnées de la procédure, il peut considérer que certains états de terminaison réalisent certains objectifs et d'autres non.

- Certaines propriétés avancées du calcul des événements permettent de représenter des changements non-déterministes de valeur de vérité de fluent. Ces techniques ne sont pas abordées dans ce texte. Le lecteur intéressé peut consulter l'ouvrage de référence de Mueller [55] à cet effet.

Il est possible que plusieurs procédures satisfassent un même objectif et qu'une seule procédure en satisfasse plusieurs. Dans la post-condition de la procédure d'imagerie par résonnance magnétique nucléaire du sein, on constate que deux objectifs sont atteints au terme de cette procédure : la tumeur est localisée et sa taille est mesurée.

$$\neg HoldsAt(Terminé(IRMSein, patient), t-1) \wedge$$
$$HoldsAt(Terminée(IRMSein, patient), t) \Rightarrow$$
$$Happens(Satisfaction(TumeurLocalisée, patient, ICCancerDuSein), t) \wedge$$
$$Happens(Satisfaction(ExtensionLocaleMesurée, patient, ICCancerDuSein), t) \wedge \ldots$$

6.13 Représentation des objectifs

Nous avons déjà souligné les avantages de représenter explicitement les objectifs d'un processus lors de sa modélisation. La méthodologie KAOS nous suggère de représenter les objectifs sous forme d'expressions en logique temporelle linéaire ou LTL (voir section 3.7.1). Ces formules font références à des états souhaitables du système dans le passé, le présent ou le futur. Cette formalisation guide l'analyste car elle lui permet d'appliquer des motifs de raffinements stéréotypés dont la validité a été prouvée une fois pour toute.

Les formules LTL de KAOS se traduisent aisément en formules du calcul événementiel. Voici une traduction de la sémantique de trois des huit opérateurs temporels de la LTL :

LTL	Calcul événementiel
$\Diamond p$	$\exists t : HoldsAt(p, t) \wedge Présent < t$
$\Box p$	$HoldsAt(p, t) \wedge Présent < t$
$\circ p$	$HoldsAt(p, Présent + 1)$

Présent est une fonction qui retourne l'instant présent du système. Pour rappel, dans ce texte, les variables qui ne font pas l'objet d'une quantification explicite sont implicitement quantifiées universellement (voir III). Les autres opérateurs de la LTL se traduisent aussi facilement. Ces objectifs sont exprimés par rapport à un instant présent, quel qu'il soit. Il doivent donc être satisfaits à tous les instants.

Dans ce travail, nous adoptons un point de vue différent. Nous considérons un objectif comme une certaine entité qui peut-être satisfaite ou pas à un certain instant t. L'occurrence d'un événement ou un changement d'état dans le processus peut satisfaire ou au contraire, ne plus satisfaire un objectif. Il nous semble que cette façon de voir les choses facilite certaines représentations comme par exemple les relations de précédences entre objectifs dont nous avons besoin dans nos modèles mais qui sont absentes du framework KAOS.

Dans le contexte des processus de soins, on représente un objectif par deux fluents $Satisfait(objectif, patient, processus))$ et $Insatisfait(objectif, patient, processus)$

objectif est le l'intitulé de l'objectif (ex. : *PatientTraité*), *patient* est l'identifiant du patient concerné par cet objectif (ex ; : *Dupont*, 74072303406 (numéro de carte SIS)) et *processus* est l'identifiant du processus dans lequel cet objectif est d'application (ex : *ItinéraireCliniqueCancerSein*). Deux événements sont responsables des changements d'état d'un objectif : $Satisfaction$ et $Insatisfaction$. Les axiomes suivants lient ces événements à ces états :

$Initiates(Satisfaction(objectif, patient, processus), Satisfait(objectif, patient, processus))$
$Terminates(Satisfaction(objectif, patient, processus), Insatisfait(objectif, patient, processus))$
$Initiates(Insatisfaction(objectif, patient, processus), Insatisfait(objectif, patient, processus))$
$Terminates(Insatisfaction(objectif, patient, processus), Satisfaction(objectifs, patient, processus))$

Nous avons vu à la section précédente comment l'exécution des procédures influençait la satisfaction des objectifs. Les procédures ne sont pas les seules à avoir une influence sur les objectifs. L'état du patient ou de l'environnement peuvent également les influencer. Par exemple, si la tension artérielle systolique du patient est représentée par trois fluents $Paramètre(patient, TAS, Basse)$, $Paramètre(patient, TAS, Normale)$ et $Paramètre(patient, TAS, Haute)$, l'axiome suivant encode l'effet de ce paramètre sur l'objectif $TASNormalisée$:

$$\neg HoldsAt(Paramètre(patient, TAS, Normale), t-1) \wedge$$
$$HoldsAt(Paramètre(patient, TAS, Normale), t) \Rightarrow$$
$$Happens(Satisfaction(TASNormalisée, patient, ICHTAChronique), t)$$

Nous avons déjà constaté que les objectifs trouvent leur place dans une hiérarchie de raffinement ET-OU. Les sections suivantes proposent une représentation de ces relations en logique événementielle.

6.13.1 Raffinements ET

La figure 6.20 représente un fragment du modèle d'objectifs de l'itinéraire clinique de prise en charge du cancer du sein. Ce raffinement ET peut se représenter à l'aide des axiomes suivants :

$$\neg HoldsAt(Satisfait(SymptômesContrôlés, p, ICCancerSein), t) \wedge$$
$$HoldsAt(Satisfait(DouleurContrôlée, p, ICCancerSein), t) \wedge$$
$$HoldsAt(Satisfait(DépressionContrôlée, p, ICCancerSein), t) \wedge$$
$$HoldsAt(Satisfait(NauséeContrôlée, p, ICCancerSein), t)$$
$$\Rightarrow Happens(Satisfaction(SymptômesContrôlés, p, ICCancerSein), t)$$

$$HoldsAt(Satisfait(SymptômesContrôlés, p, ICCancerSein), t) \wedge$$
$$((HoldsAt(Satisfait(DouleurContrôlée, p, ICCancerSein), t-1) \wedge$$
$$\neg HoldsAt(Satisfait(DouleurContrôlée, p, ICCancerSein), t)) \vee$$
$$(HoldsAt(Satisfait(DépressionContrôlée, p, ICCancerSein), t-1) \wedge$$
$$\neg HoldsAt(Satisfait(DépressionContrôlée, p, ICCancerSein), t)) \vee$$
$$(HoldsAt(Satisfait(NauséeContrôlée, p, ICCancerSein), t-1) \wedge$$
$$\neg HoldsAt(Satisfait(NauséeContrôlée, p, ICCancerSein), t))) \Rightarrow$$
$$Happens(Insatisfaction(SymptômesContrôlés, p, ICCancerSein), t)$$

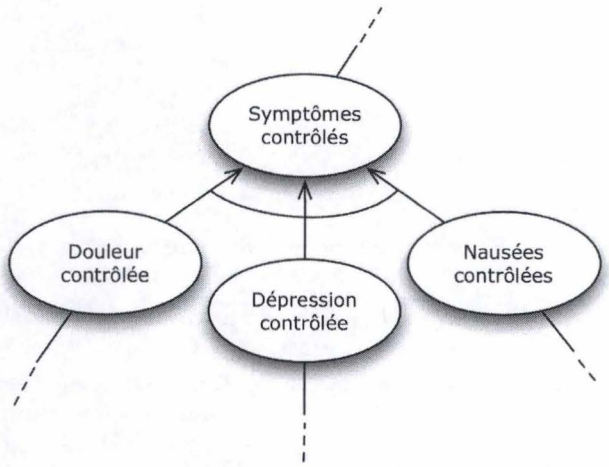

FIG. 6.20 – Exemple de raffinement-ET

L'objectif père est satisfait tant que ses objectifs-fils sont tous satisfaits.

Dans cet exemple, les objectifs-fils sont relatifs au contrôle de symptômes susceptibles d'évoluer au cours du temps. Par exemple, en fonction du traitement prescrit, la douleur doit-être réévaluée tous les jours, les nausées tous les 2 jours et la dépression toutes les trois semaines. Ces durées résultent de paramètres médicaux internes aux procédures, dans ce cas ci, la durée d'action des traitements.

Il existe plusieurs possibilités pour représenter le caractère temporaire de la satisfaction de ces objectifs. Soit on lie l'insatisfaction de l'objectif à un état particulier du patient :

$\neg HoldsAt(Symptôme(patient, Nausée), t-1) \wedge$

$HoldsAt(Symptôme(patient, Nausée), t) \Rightarrow$

$Happens(Insatisfaction(NauséeContrôlée, patient, ICCancerDuSein), t)$

La conséquence de cette représentation est que le patient doit nécessairement transiter par un état nauséeux pour avoir une chance de se faire traiter. Or, la notion de contrôle des symptômes contient une dimension d'anticipation de ceux-ci qui n'apparaît plus dans le modèle. En général, nous pensons qu'il faut éviter de faire reposer la satisfaction des objectifs sur des états du patient. Cette règle souffre bien entendu d'exceptions.

Une façon préférable de procéder est d'indiquer le caractère temporaire de l'objectif dans les post-conditions de la procédure qui le satisfait (la procédure, simple, de contrôle des nausées est présentée à la figure 6.21) :

$\Rightarrow Happens(Satisfaction(NauséeContrôlée, patient, ICCancerDuSein), t) \wedge$

$Happens(Insatisfaction(DouleurContrôlée, p, ICCancerDuSein), t+2)$: la procédure satisfait l'objectif pendant deux jours.

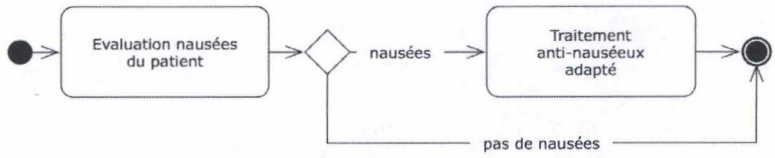

FIG. 6.21 – Procédure de contrôle des nausées d'un patient

Si plusieurs procédurent concourent à la satisfaction du même objectif, on peut ainsi représenter que certaines ont des effets plus durables que d'autres.

Enfin, une dernière possibilité consiste à utiliser des caractéristiques avancées du calcul des événements qui permettent de représenter certains phénomènes continus comme par exemple d'inférer des effets de médicaments en fonction de leurs caractéristiques pharmacocinétiques. Ces notions ne seront pas abordées dans ce texte. L'auteur renvoie le lecteur intéressé à l'ouvrage de référence de Mueller [55].

6.13.2 Relations de précédence entre objectifs

Le raffinement-ET représenté à la figure 6.11 pose un problème par rapport à la définition qui en a été donnée à la section précédente. Il existe en effet une précédence temporelle à respecter dans la satisfaction de ces objectifs. Cela n'a bien évidemment aucun sens que le suivi du patient soit assuré pour un cancer dont le diagnostic n'a pas encore été posé. Ces relations de précédences sont liées à des dépendances de natures diverses qui existent entre les étapes d'un workflow. Par exemple, entre le diagnostic et le traitement, il existe une dépendance d'informations ou de connaissance de l'état du patient. Dans le modèle d'objectifs, nous choisissons de représenter ces relations de précédence temporelle par des flèches pointillées. L'objectif situé à l'origine de la flèche doit être satisfait avant l'objectif situé à l'extrémité de la flèche. Les axiomes suivants représentent formellement cette précédence dans le raffinement.

$\neg HoldsAt(Satisfait(PatientPrisEnCharge, p, ICCancerSein), t) \wedge$
$\neg HoldsAt(Satisfait(PathologieEvaluée, patient, ICCancerSein), t1 - 1) \wedge$
$HoldsAt(Satisfait(PathologieEvaluée, patient, ICCancerSein), t1) \wedge$
$\neg HoldsAt(Satisfait(PatientTraité, patient, ICCancerSein), t2 - 1) \wedge$
$HoldsAt(Satisfait(PatientTraité, patient, ICCancerSein), t2) \wedge$
$\neg HoldsAt(Satisfait(PatientSuivi, patient, ICCancerSein), t3 - 1) \wedge$
$HoldsAt(Satisfait(PatientSuivi, patient, ICCancerSein), t3) \wedge$
$((t1 < t2 \wedge t2 < t3) \vee (t2 < t1 \wedge t1 < t3))$
$\Rightarrow Happens(Satisfaction(PatientPrisEnCharge, patient, ICCancerSein), t3)$

$HoldsAt(Satisfait(PatientPrisEnCharge, p, ICCancerSein), t) \wedge$
$((HoldsAt(Satisfait(PathologieEvaluée, p, ICCancerSein), t - 1) \wedge$
$\neg HoldsAt(Satisfait(PathologieEvaluée, p, ICCancerSein), t)) \vee$
$(HoldsAt(Satisfait(PatientTraité, p, ICCancerSein), t - 1) \wedge$
$\neg HoldsAt(Satisfait(PatientTraité, p, ICCancerSein), t)) \vee$

$(HoldsAt(Satisfait(PatientSuivi, p, ICCancerSein), t-1) \wedge$
$\neg HoldsAt(Satisfait(PatientSuivi, p, ICCancerSein), t))) \Rightarrow$
$Happens(Insatisfaction(PatientPrisEnCharge, p, ICCancerSein), t)$

Cet exemple illustre une fois de plus la nécessité de disposer d'un outil de génération de ces formules à partir d'un modèle graphique d'objectifs.

6.13.3 Raffinements OU

Le raffinement-OU n'appelle pas beaucoup de commentaires. La figure 6.22 se représente de la façon suivante :

$\neg HoldsAt(Satisfait(TumeurRetirée, p, ICCancerDuSein), t) \wedge$
$(HoldsAt(Satisfait(TumorectomiePratiquée, p, ICCancerDuSein), t) \vee$
$HoldsAt(Satisfait(QuadrantectomiePratiquée, p, ICCancerDuSein), t) \vee$
$HoldsAt(Satisfait(MastectomiePratiquée, p, ICCancerDuSein), t))$
$\Rightarrow Happens(Satisfaction(TumeurRetirée, p, ICCancerDuSein), t)$

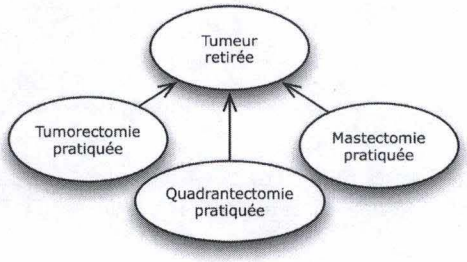

FIG. 6.22 – Exemple de raffinement-OU

6.14 Inférence de scénarios d'exécution de workflow

Le processus d'ingénierie de processus, brièvement suggéré dans ce chapitre, produit deux modèles distincts : un graphe de raffinement d'objectifs et un modèle opérationnel constitué d'un ensemble de procédures. Si cette "librairie" de procédures offre des caractéristiques utiles d'adaptabilité et de flexibilité, il semble que nous ayons perdu le fil du workflow complet. Nous avons déjà signalé à plusieurs reprises dans ce travail, que nous comptions utiliser le raisonnement abductif pour générer des scénarios d'événements correspondant au démarrage et à la fin des procédures et faire usage de ces informations pour influencer l'exécution du workflow sur le terrain.

La figure 6.23 présente une vision schématique du raisonnement abductif. Un état de départ et un objectif sont fournis au moteur de raisonnement. A partir des axiomes du domaine, le moteur génère les scénarios d'événements compatibles.

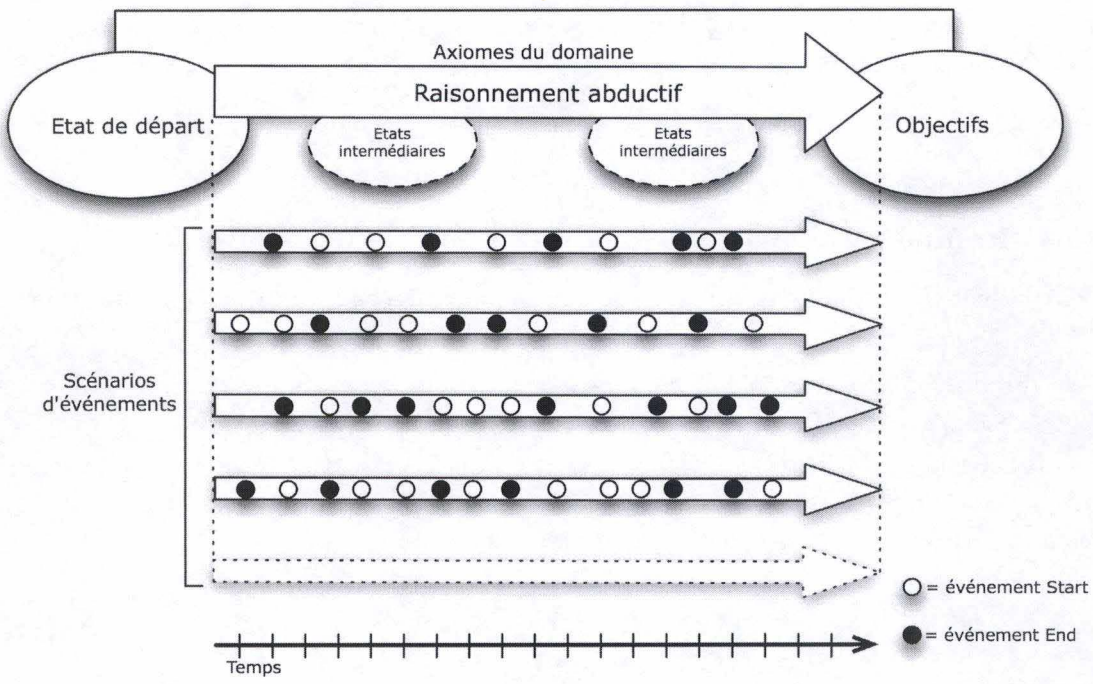

FIG. 6.23 – Représentation schématique du raisonnement abductif

L'état de départ est une conjonction de prédicats $HoldsAt$ valables à l'instant initial 0 du processus. L'expression suivante est un état de départ typique qui indique qu'à l'instant initial, aucun objectif n'est atteint :

$$HoldsAt(Insatisfait(objectif, Dupont, ICCancerSein), 0)$$

$objectif$ est une variable implicitement universellement quantifiée qui vaut pour tous les objectifs du processus. $Dupont$ est la constante qui désigne le patient au processus duquel nous nous intéressons. Dans certaines situations, on peut considérer qu'une partie des objectifs sont déjà atteints à l'état de départ. C'est le cas pour des patients transférés d'un hôpital à l'autre et qui démarrent le traitement avec un diagnostic déjà posé.

L'objectif est également une conjonction de prédicats $HoldsAt$ valables à un instant t_{max} du processus. La séquence d'instants $0 \rightarrow t_{max}$ représente la période de temps que l'on désire couvrir lors de l'inférence. Dans le cas d'un processus d'une durée maximale estimée à 30 jours et dont la résolution temporelle est de 1 jour (chaque entier positif représente 1 jour), t_{max} doit être au moins égal à 30 pour couvrir toute la durée du processus.

L'expression suivante est un objectif typique qui exprime qu'à l'instant maximal considéré, l'objectif racine du graphe de raffinement ET-OU doit être atteint à l'instant maximal considéré t_{max}= 200 :

$$HoldsAt(Satisfait(PatientPrisEnCharge, Dupont, ICCancerSein), 200)$$

Comme il s'agit d'un *graphe* de raffinement ET-OU, il est possible qu'il existe plusieurs objectifs racines à atteindre.

Entre l'état de départ et l'objectif, ont peut également imposer certains états intermédiaires. C'est en général une bonne pratique qui permet de limiter le nombre de scénarios inférés. Ces états intermédiaires prennent également la forme de conjonctions de prédicats $HoldsAt$ valables à des instants $< t_{max}$:

$$\neg HoldsAt(Satisfait(PathologieEvaluée,\ Dupont,\ ICCancerSein),\ t-1) \wedge$$
$$HoldsAt(Satisfait(PathologieEvaluée,\ Dupont,\ ICCancerSein),\ t) \wedge$$
$$t < 16$$

Dans ce cas on ne s'intéresse qu'aux modèles qui assurent que le diagnostic est posé dans les 15 premiers jours du processus.

A partir de ces états du processus, des axiomes du calcul des événements et des axiomes du domaine qui décrivent à la fois le modèle intentionnel et le modèle opérationnel du processus, le moteur de raisonnement va générer la série complète des scénarios d'événements compatibles avec le modèle. Chaque scénario est une succession temporisée d'événements. Chaque événement est représenté par un prédicat $Happens(événement,\ t_{occurrence})$. Ce sont ces événements qui modifient les valeurs de vérité des fluents dans les modèles.

Dans ces scénarios se sont principalement les événements $StartProcédure$ et $EndProcédure$ qui nous intéressent. Ils signent le démarrage des procédures. Les événements $Start$ et End seront également présents à condition que la spécification contienne une description de la structure interne des procédures en terme de succession d'activités coordonnées.

6.14.1 Planification des procédures

A partir de l'information contenue dans ces scénarios, il est théoriquement possible de construire un planning des activités du processus. Néanmoins, les procédures contiennent des noeuds décisionnels qui induisent de nombreux chemins d'exécution possibles. A cause de cette indétermination, la construction d'un planning au niveau des activités a peu d'intérêt. Ce planning peut néanmoins être construit au niveau des procédures ce qui apporte déjà une information très utile.

Soit S l'ensemble de tous les scénarios $s_{1...n}$ générés par le moteur de raisonnement. Soit P l'ensemble de toutes les procédures $p_{1...m}$ contenues dans la spécification du processus. Chaque scénario s_i contient un exemplaire d'un événement $StartProcédure$ d'une procédure p_j noté $Start(p_j)$ et d'un événement $EndProcédure$ de cette même procédure noté $End(p_j)$.

$t_{Start\,p_j,i}$ désigne l'instant d'occurence de l'événement $Start(p_j)$ dans le scénario s_i.

$t_{End\,p_j,i}$ désigne l'instant d'occurence de l'événement $End(p_j)$ dans le scénario s_i :

$min_{i=1...n}(t_{Start\,p_j,i})$ est l'instant minimum de démarrage possible de la procédure p_j dans le processus.

$max_{i=1...n}(t_{Start\,p_j,i})$ est l'instant maximum de démarrage possible de la procédure p_j dans le processus.

$min_{i=1...n}(t_{End\,p_j,i})$ est l'instant minimum de terminaison possible de la procédure p_j dans le processus.

$max_{i=1...n}(t_{End\,p_j,i})$ est l'instant maximum de terminaison possible de la procédure p_j dans le processus.

On peut calculer ces quatre instants pour toutes les procédures du processus.

Ce raisonnement est valable moyennant l'adoption des conventions de nommage adéquates pour individualiser les instances d'exécution des procédures (voir section 5.12.2).

A partir des valeurs minimales et maximales de démarrage et de terminaison des procédures, il est possible d'offrir un support aux agents responsables sur le terrain en leur proposant un service de gestion des procédures.

Lorsque l'instant minimum de démarrage d'une procédure est atteint, le système peut avertir les membres du rôle responsable de la nécessité de mettre cette procédure en oeuvre. La responsabilité de mener à bien la procédure est attribuée à l'un d'entre eux.

Lorsque l'instant maximum de démarrage d'une procédure est atteint, cette procédure peut être "imposée" à un agent membre du rôle selon une heuristique à définir (files de priorités, estimation de la charge de travail, ...).

Lorsque l'on atteint l'instant minimum de fin d'une procédure, l'agent à qui cette procédure a été attribuée a la possibilité de la déclarer terminée en fonction de ce qui a été accompli sur le terrain.

Lorsqu'une certaine fraction de l'intervalle de temps, entre le moment où la procédure a été attribuée à un agent et l'instant maximum de terminaison de cette procédure s'est écoulée, le système peut envoyer un avertissement à l'agent responsable. Il peut également afficher en direct le temps restant pour mener à bien la procédure.

Les violations de la planification établie par le système sont notées comme des déviations du processus. La raison de ces déviations, surtout si elles sont systématiques, peut faire l'objet d'une analyse ultérieure qui permettra de décider si le processus doit être modifié en vue de son amélioration.

L'attribution de la réalisation des procédures aux agents nécessite une définition claire des responsabilités et des compétences de chacun d'eux. A la section 2.2.4, nous avons relevé que ces notions sont d'une importance particulière en médecine à cause d'enjeux médico-légaux et déontologiques sous-jacents. La représentation précise de ces concepts ne sera pas abordée dans ce texte. Le lecteur intéressé pourra consulter l'article de Koubarakis [41] qui aborde cet aspect.

Nous avons également constaté à la section 6.10 que plusieurs procédures peuvent satisfaire le même objectif et que certaines métadonnées peuvent leur être ajoutées. Une utilisation de ces métadonnées pourrait-être le filtrage des scénarios obtenus par raisonnement abductif. Le système serait à même d'identifier les scénarios les moins coûteux, les moins risqués, ... ou les scénarios compatibles avec une combinaison de certains de ces critères. Utilisé de cette façon, le système pourrait fournir une forme d'aide à la décision.

6.15 Vérification des propriétés du modèle

La vérification d'un système (dans notre cas, un processus de soins) consiste à prouver que ce système possède bien certaines propriétés souhaitées. Deux stratégies de vérification sont envisageables :

- le test
- la vérification formelle

On peut exprimer de façon procédurale ces propriétés sous forme de cas de test. Leur vérification consistera à soumettre le modèle de processus à une campagne de tests. Pour beaucoup de systèmes

informatiques, une sélection soigneuse de cas de test suffit à s'assurer que l'on rencontre les exigences voulues. Néanmoins, l'efficacité de la campagne de tests dépend essentiellement de la bonne sélection des cas de test. Une campagne de tests bien menée peut confirmer la présence d'erreurs mais est incapable de prouver l'absence d'erreurs.

Il existe des systèmes qui nécessitent un niveau de sûreté tellement élevé que les tests au sens traditionnel du terme ne sont pas suffisants. Il s'agit de systèmes critiques dans lesquels des erreurs peuvent occasionner un préjudice humain ou économique élevé. Les processus médicaux sont considérés comme des systèmes critiques. Ils nécessitent de pousser plus loin leur vérification.

Une stratégie différente consiste à exprimer les propriétés souhaitées de façon déclarative, par exemple sous la forme de formules logiques. Dans ce cas, la vérification du processus consistera à utiliser des outils de preuve automatique (theorem prover) ou de vérification de modèles (model checker). Ces outils sont capables de prouver que la spécification du processus satisfait bien les propriétés exprimées. Néanmoins, ils ne fonctionnent pas sur les processus réels mais sur des modèles, exprimés dans un formalisme adéquat. Le risque est bien entendu que le modèle ne soit pas un reflet fidèle de la réalité.

Lorsqu'un processus métier est représenté à l'aide de formules logiques, un outil peut en dériver automatiquement des conclusions par répétition automatique des règles d'inférence de la logique utilisée. On peut utiliser ce mécanisme pour vérifier la consistance du modèle. Une stratégie de tests consisterait à imposer des conditions non souhaitées, et voir si le système de raisonnement trouve des modèles compatibles avec ces conditions.

Par exemple, sachant que deux molécules utilisées en chimiothérapie ont des interactions que l'on souhaite à tout prix éviter, on peut ajouter les prédicats suivants à la spécification du processus de soins :

$$HoldsAt(Traitement(patient,\ Paclitaxel),\ t) \wedge HoldsAt(Traitement(patient,\ Cisplatine),\ t)$$

Cette formule, qui représente la condition que l'on veut justement éviter, sélectionne les scénarios dans lesquels ces deux traitements sont donnés en même temps. Si le moteur de raisonnement ne renvoie aucun scénario, le modèle n'autorise pas que ces traitements soient donnés en même temps. Dans le cas contraire, les scénarios renvoyés contiennent le problème. Leur analyse doit permettre de définir dans quelles circonstances ce problème se pose et corriger la spécification de processus en conséquence.

L'exemple précédent vérifiait une propriété liée à un état particulier du patient. On peut également vérifier des propriétés liées aux aspects temporels du processus. Par exemple, pour éviter une évolution trop rapide du cancer en cours de traitement, on souhaite que la tumeur primitive soit retirée dans les six semaines (42 jours) après l'obtention du diagnostic. La formule suivante exprime cette propriété :

$$\begin{aligned}
&\neg HoldsAt(Satisfait(PathologieEvaluée,\ patient,\ ICCancerSein),\ t_1 - 1) \wedge \\
&HoldsAt(Satisfait(PathologieEvaluée,\ patient,\ ICCancerSein),\ t_1) \wedge \\
&\neg HoldsAt(Satisfait(TumeurRetirée,\ patient,\ ICCancerSein),\ t_2 - 1) \wedge \\
&HoldsAt(Satisfait(TumeurRetirée,\ patient,\ ICCancerSein),\ t_2) \wedge \\
&t_2 < 42 - t_1
\end{aligned}$$

Si le moteur de raisonnement ne renvoie aucun scénario, alors, le processus modélisé n'est pas conforme à la propriété. La localisation du problème est ici plus délicate car on n'obtient pas d'information. Une stratégie possible serait d'affaiblir progressivement la contrainte (ici en augmentant le délai) jusqu'à obtenir des scénarios dont l'analyse est susceptible de nous aider.

6.16 Résumé

Si différents domaines de l'informatique ont accordé une place à la représentation des objectifs, dans le domaine des workflows, peu de travaux ont été consacrés à ce sujet. Dans ce chapitre, les concepts d'agents, d'objectifs comportementaux et d'objectifs souples sont définis. Les grandes catégories d'objectifs fonctionnels et non-fonctionnels sont également évoquées ainsi que leurs pendants dans le domaine médical. Le balayage systématique de ces catégories pendant le processus d'élicitation permet de découvrir des objectifs supplémentaires. En partant d'un cas réel, la démonstration est faite qu'il existe bel et bien des objectifs dans les processus de soins et que leur mise en évidence induit un découpage du processus en grandes étapes fonctionnelles. Ces objectifs trouvent naturellement leur place dans une hiérarchie de relations de contribution. Cette hiérarchie est en réalité un graphe de raffinement ET-OU et il existe des heuristiques simples qui permettent de le construire de façon systématique. C'est la réalisation des objectifs par au moins une procédure sous la responsabilité d'un rôle bien défini qui constitue le critère de terminaison du raffinement. La construction d'un modèle d'objectifs comme abstraction centrale pour modéliser des workflows offre une série d'avantages dont une meilleure évolutivité des modèles. Le calcul des événements permet de représenter précisément les procédures qui composent un workflow, les objectifs qu'elles contribuent à atteindre et leur graphe de raffinement. L'application du raisonnement abductif sur cette spécification mixte (intentionnelle et opérationnelle) permet de générer des scénarios d'appels de procédures compatibles avec la spécification. Ces scénarios peuvent être utilisés pour planifier l'exécution des procédures dans la réalité. Le fait de disposer d'une spécification formelle du processus permet de vérifier sa conformité par rapport à certaines propriétés liées notamment à la sécurité du patient.

Les informations utilisées dans l'introduction aux approches orientées-objectifs proviennent des pages 280 à 283 de l'ouvrage de van Lamsweerde [96]. Les quelques indications à propos de la modélisation d'organisations complètes proviennent de l'article de Koubarakis [41].

Les sections 6.2 à 6.4 contiennent des défintions adaptées du framework KAOS et issues du chapitre 7 de l'ouvrage de van Lamsweerde [96].

La catégorisation des objectifs médicaux de la section 6.5 est adaptée de l'article de John Fox [30].

Le concept de raffinement ET-OU d'objectifs et la suggestion de l'heuristique "pourquoi, comment ?" est due à van Lamsweerde [96].

Chapitre 7

Conclusions et travaux futurs

Dans l'introduction de ce texte, nous nous étions fixé quatre principaux objectifs à mener à bien (section 1.2) :

1. explorer le domaine des workflows et des principaux formalismes utilisables pour les représenter, en particulier dans le contexte des processus de soins hospitaliers.
2. évaluer l'apport méthodologique de la construction d'un modèle d'objectifs en parallèle à la construction d'un modèle de workflow.
3. définir une technique de représentation explicite des objectifs au même titre que les activités dans un modèle de processus.
4. découvrir un mécanisme qui exploite ce modèle intentionnel d'objectifs pour influencer l'enchaînement des activités du processus.

Il nous semble qu'au terme de ce travail, nous ayons progressé dans la réalisation de chacun d'entre eux.

Aux chapitres 2, 3 et 4, nous avons survolés les principaux langages de workflow disponibles et mis en évidence certaines caractéristiques particulières des processus de soins. Nous avons notamment reconnu le besoin de rigueur dans leur définition et de flexibilté opérationnelle dans leur exécution. Cette partie du travail contribue à la satisfaction de notre premier objectif.

A ce stade, nous sommes parvenus à la conclusion qu'un langage de workflow propre à représenter des processus cliniques doit sans doute se baser sur une forme de logique. Il doit également permettre de spécifier les objectifs du processus dans un modèle indépendant et synergique de celui des activités. Notre choix s'est porté sur le calcul événementiel.

Le chapitre 5 a apporté la preuve que le calcul événementiel permettait de représenter précisément la succession des activités coordonnées d'un workflow, c'est à dire son modèle opérationnel. Dans le chapitre 6, nous avons montré que le même formalisme était utilisable pour représenter le modèle intentionnel du processus sous la forme d'un graphe de raffinement ET-OU d'objectifs. Nous avons également explicité la nature des relations qui existent entre ces deux modèles : les activités individuelles ou les procédures complètes satisfont les objectifs au cours du déroulement du processus. Il peut s'agir d'une satisfaction directe : la fin de l'activité ou de la procédure correspond à la satisfaction inconditionnelle de l'objectif. Il peut également s'agir d'une statisfaction indirecte : l'activité ou la procédure exerce une influence sur l'état du patient ou de l'environnement du processus. C'est ce changement d'état ou une combinaison de changements d'états qui satisfait l'objectif. C'est également au chapitre 6 que nous avons relevé qu'il existait au moins deux niveaux de représentation du modèle opérationnel : une vision de haut niveau, qui ne considère que le descriptif des pré- et

de post-conditions de procédures, et une vision de bas niveau qui représente explicitement chaque activité.

Cette partie du mémoire nous a permis de progresser dans la satisfaction du troisième objectif que nous nous étions fixé.

Le chapitre 5 à présenté les trois principaux modes de raisonnement supportés par le calcul événementiel : le raisonnement déductif, le raisonnement inductif et le raisonnement abductif. Dans le chapitre 6, nous avons montré de quelle façon utiliser le raisonnement abductif pour générer un planning d'exécution des procédures du processus à partir de son graphe d'objectifs. Nous avons également suggéré que le planificateur obtenu puisse apporter une certaine forme d'aide à la décision s'il prend en compte des métadonnées associées aux procédures. La planification réalisée au niveau des activités individuelles n'a pas beaucoup de sens eu égard à la présence de noeuds décisionnels qui rendent le processus de soins non-déterministe. Selon nous, cette partie du mémoire contribue clairement à notre objectif numéro 4.

En parallèle de cette approche théorique, ce travail a été l'occasion pour l'auteur de mettre en pratique le principe de construction d'un modèle d'objectifs pour guider la conception du workflow d'un processus de soins. Cette expérimentation a été menée à petite échelle avec la future coordinatrice d'un itinéraire clinique de prise en charge du cancer du sein, une adjointe à la direction de l'hôpital, deux médecins radiothérapeutes et oncologues et quelques membres du personnel soignant. Ce travail, encore incomplet à l'heure actuelle, a permis de tirer un certain nombre d'enseignements méthodologiques :

- L'élicitation des objectifs selon un arbre de raffinement ET-OU en utilisant des heuristiques simples est un mécanisme naturel qui fonctionne bien pour peu qu'il soit encadré par un analyste expérimenté. Dans le cas qui nous intéresse, le modèle d'objectifs était créé au cours de séances de travail interactives avec des Post-its apposés sur un tableau blanc.
- Le modèle obtenu est plus modulaire, mieux transplantable et à priori plus évolutif. Il contient sa propre justification.
- Les objectifs rendent visible la contribution respective de chaque procédure de bas niveau à la réalisation d'objectifs thérapeutiques de haut niveau. Ce point est très important et incite à rendre public le modèle d'objectifs comme facteur de motivation des soignants de terrain, voire du patient.
- La réalisation des objectifs est un critère précis d'avancement du workflow. Dans le processus réel, même en l'absence de support informatique, on peut associer la réalisation des objectifs à la mesure d'indices de performance ou de qualité. Cette approche est, en outre, particulièrement en phase avec la philosophie des itinéraires cliniques.

Néanmoins, une fois le modèle terminé, le fil d'exécution du workflow complet est d'une certaine manière "noyé dans les objectifs". Il est bien entendu possible, lors d'une étape ultérieure, de replacer chacune des procédures dans un modèle opérationnel global. L'auteur pense que l'utilisation d'un mécanisme abductif de plannification est plus élégant et surtout plus flexible mais nécessite un support informatique adéquat qui n'existe pas à l'heure actuelle.

Ces enseignements apportent une contribution à la réalisation de notre deuxième objectif.

7.1 Avantages de l'approche proposée

Ce travail est une première approche encourageante de la construction de modèles de processus de soins à la fois intentionnels et opérationnels :

- le formalisme choisi, à savoir le calcul événementiel, est bien adapté à la spécification précise de ces deux dimensions complémentaires.
- la disponibilité d'outils de raisonnement automatique qui supportent le calcul événementiel permet de valider et de vérifier formellement les spécifications de processus.
- le raisonnement adbuctif constitue un mécanisme de planification des procédures, offert "pour rien" par les outils de raisonnement.
- la construction simultanée du modèle d'objectifs apporte une aide méthodologique à la construction du modèle opérationnel. D'autres méthodologies, notamment celles proposées pour élaborer des itinéraires cliniques sont néanmoins toujours utilisables.
- A condition de disposer de l'environnement informatique adéquat pour les exécuter, les modèles obtenus offrent à priori une plus grande flexibilité d'exécution. Ce point est aujourd'hui suputatif et n'a pas été validé dans la réalité.

7.2 Limitations de cette approche

Cette première approche présente néanmoins un certain nombre de problèmes :

- Le calcul événementiel est un formalisme d'assez bas niveau pour exprimer des workflows. Les spécifications a écrire atteignent vite une longueur importante. Sans outil adéquat de plus haut niveau, l'utilisation du calcul événementiel "nu" semble peu praticable pour décrire des processus de taille réelle.
- Le support de l'exécution simultanée de plusieurs instances du même processus, de la même procédure ou de la même activité nécessite, pour être praticable, la génération et la gestion intelligente de noms d'instances.
- l'outil de raisonnement implémenté par Mueller et utilisé par l'auteur pour valider l'approche offre un support limité à l'écriture des spécifications. Certaines spécifications incomplètes ou erronées provoquent un blocage du processus de raisonnement dont il est difficile de déterminer la cause.
- L'approche proposée reste pour l'instant plus théorique que pratique. Elle le restera tant qu'elle n'aura pas été utilisée pour représenter un processus de soins d'une taille significative.
- Une mesure de la performance de l'outil de raisonnement sur des spécifications de taille réelle serait également nécessaire. Si l'itinéraire clinique d'un patient doit être replanifié régulièrement (par exemple à chaque nouvelle hospitalisation ou à la fin de chaque procédure), un niveau de performance minimal est souhaitable.

7.3 Travaux futurs

Ce mémoire pourrait-être le point de départ d'une série de développements et de recherches intéressants. Du point de vue du développement, :

- l'introduction d'un moteur d'inférence abductif dans un interpréteur de workflow pour supporter l'exécution de modèles de workflows mixtes. Ce pourrait être un développement à envisager dans le cadre du projet APCOS évoqué dans l'introduction du mémoire.

- l'extension d'un outil de modélisation graphique de type UML pour générer une spécification de processus métier en calcul événementiel à partir de modèles de plus haut niveau. La description des procédures pourrait être générée à partir de diagrammes d'activités. La description du modèle d'objectifs pourrait l'être à partir d'un modèle de classes adapté à l'aide d'un profil UML dédié. Il existe déjà un profil UML qui supporte le métamodèle KAOS [33] et qui pourrait être utilisé moyennant quelques adaptations.

Du point de vue de la recherche :

- Le premier travail de recherche a effectuer serait de réaliser la modélisation complète d'un processus réel, aidé en cela par l'outil construit à la section précédente.
- Nous avons signalé à plusieurs reprises dans ce texte que le calcul des événements pouvait être utilisé pour représenter des changements continus dans l'environnement, des phénomènes non-déterministes ou des effets combinés d'événements. Il serait intéressant de voir si l'utilisation de ces techniques particulières apporte quelque chose d'utile à la modélsation de processus, particulièrement médicaux.
- KAOS offre un mécanisme de dérivation systématique d'objectifs de haut niveau en objectifs plus opérationnels. Ce mécanisme repose sur des motifs de raffinement qui ont été logiquement prouvé une fois pour toute. En nous écartant de la sémantique de KAOS, notamment en tolérant des objectifs dont la satisfaction change au fil du temps, nous nous coupons de la possibilité d'utiliser ces motifs. Il serait intéressant de voir si cette notion de motifs de raffinement peut être transposée au calcul des événements pour automatiser en partie ou vérifier le raffinement proposé.
- La relation de précédence entre objectifs est en général liée à des dépendances entre des étapes du workflow (dépendance de données, dépendance de ressources, ...). Une modélisation explicite de ces dépendances rendrait-elle caduque ou pas cette relation de précédence ?
- Nous avons dit qu'il peut exister des relations de priorité entre objectifs. Dans ce mémoire, nous ne les avons pas représentées. Une recherche intéressante pourrait-être de les représenter explicitement et de voir si elles sont réellement utiles dans la modélisation de processus.
- KAOS propose un mécanisme d'analyse de risque sur les modèles d'objectifs à travers la notion d'obstacle. Un obstacle est une condition rajoutée au modèle et qui empêche la satisfaction d'un objectif particulier. D'autres objectifs peuvent être ensuite rajoutés pour tenter de contourner l'obstacle. Il serait intéressant de voir si ce type d'analyse est transposable en calcul événementiel. Ce point est d'un intérêt particulier pour des processus médicaux considérés comme critiques.

Troisième partie

Annexes

Bibliographie

[1] Crossing the Quality Chasm : A New Health System for the 21st Century - American Institute of Medicine, 2001

[2] A.I. Anton, M.W. McCracken, and C. Potts. Goal decomposition and scenario analysis in business process reengineering. In Proceedings of CAISE'94 , pages 94- 104, 1994.

[3] L. Ardissono, A. Di Leva, G. Petrone, M. Segnan and M. Sonnessa ; Adaptive Medical Workflow Management for a Context-Dependent Home Healthcare Assistance Service ; Electronic Notes in Theoretical Computer Science ; Volume 146, Issue 1, 24 January 2006, Pages 59-68.

[4] M. Balser, C. Duelli, and W. Reif, 'Formal semantics of asbru - an overview', in Proceedings. of IDPT 2002, (june 2002).

[5] A. K. Bandara, E. C. Lupu & A. Russo, *Using Event Calculus to formalise policy specification and analysis.* Proceedings 4th IEEE workshop on polices for Distributed systems and Networks (Policy 2003), Lake Como, Italy, June 2003

[6] Bates DW, Gawande AA ; Improving safety with information technology. N Engl J Med. 2003 Jun 19 ;348(25) :2526-34.

[7] J. Bengtsson and W. Yi. *Timed automata : Semantics, algorithms, and tools.* 2004 : http ://user.it.uu.se/~yi/ps-files/chapter.ps. (retrouvé le 15 août 2009)

[8] http ://en.wikipedia.org/wiki/Backus–Naur_Form (retrouvé le 17 août 2009)

[9] A. J. Bonner, M. Kifer, Transaction logic programming, Computer Systems Research Institute Technical Report CSRI-323 (revision of CSRI-270 of 1992), University of Toronto, 1995.

[10] Boxwala AA, Peleg M, Tu S et al. GLIF3 : a representation format for sharable computer-interpretable clinical practice guidelines. J Biomed Inform. 2004 Jun ;37(3) :147-61.

[11] Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perrini, A. : *Tropos : an Agent-Oriented Software Development Methodology.* Technical Report DIT- 02-0015, University of Trento, Italy (2002)

[12] Buhler P.A., Vidal, J.M. 2005, Towards Adaptive Workflow Enactment Using Multiagent Systems. Information Technology and Management, Information Technology and Management, 6(1) :61-87.

[13] Burmeister, B., Steiert, H.-P., Bauer, T., Baumgärtel, H. 2006. Agile Processes through Goal- and Context- oriented Business Process Modeling. In Eder, J. Dustdar, S. et al. (Eds.). BPM 2006 Workshops (Wien, Austria, 2006), LNCS 4103, Springer, 215—226

[14] http ://www.cse.dmu.ac.uk/STRL/ITL/ (retrouvé le 17 août 2009)

[15] Bin Chen, George S. Avrunin, Elizabeth A. Henneman, Lori A. Clarke, Leon J. Osterweil, Philip L. Henneman, *Analyzing Medical Processes*, ACM SIGSOFT/IEEE 30th International Conference on Software Engineering (ICSE'08), Leipzig, Germany, May 2008, pp. 623-632.

[16] N.K. Cicekli and Y. Yildirim. Formalizing workflows using the event calculus. In Proc. of the 11th International Conference on Database and Expert Systems Applications (DEXA 00), 2000.

[17] N.K. Cicekli and I. Cicekli. *Formalizing the specification and execution of workflows using the event calculus*. Information Sciences, 176(15) :2227–2267, 2006.

[18] W. R. Cook and Jayadev Misra. Implementation outline of orc. December 2005 : http ://nt-appn.comp.nus.edu.sg/fm/orc/OrcImpDraft.pdf (retrouvé le 12 août 2009)

[19] W. R. Cook, Sourabh Patwardhan, and Jayadev Misra. Workflow Patterns in Orc. In Proc. of the International Conference on Coordination Models and Languages, 2006.

[20] Dam, K. H., and Winikoff, M. *Comparing Agent-Oriented Methodologies.* Proceedings of the 5th International Bi- Conference Workshop on Agent-Oriented Information Systems, 2003.

[21] C. Damas, B. Lambeau, F. Roucoux, A. van Lamsweerde Analyzing Critical Process Models through Behavior Model Synthesis Proc. ICSE'2009 : 31th International Conference on Software Engineering, Vancouver, Canada, May 16-24, 2009.

[22] A. Dardenne, S. Fickas and A. van Lamsweerde, *Goal-Directed Concept Acquisition in Requirements Elicitation.* Proceedings IWSSD-6 : Sixth International Workshop on Software Specification and Design. IEEE Computer Society Press, 1991, pp. 14-21.

[23] De Bleser, L., Vlayen, J., Vanhaecht, K., Sermeus, W. (2004). Classifying Clinical Pathways, Stud Health Technol Inform ; 110 :9-14.

[24] P.A. de Clercq, J.A. Blom, H.H. Korsten, A. Hasman, *Approaches for creating computer-interpretable guidelines that facilitate decision support*, Artif. Intell. Med. 31 (1) (2004) 1–27.

[25] Kathryn De Luc, Julian Todd ; E-pathways : Computers And the Patient's Journey Through Care ; Radcliffe Medical ; 2003.

[26] http ://www.daimi.au.dk/designCPN/ (retrouvé le 15 août 2009)

[27] EKD Enterprise Modeling Method : ftp ://ftp.dsv.su.se/users/js/ekd_user_guide_2001.pdf (retrouvé le 12 août 2009)

[28] H. Eshuis. Semantics and Verification of UML Activity Diagrams for Workflow Modelling. PhD thesis, University of Twente, Enschede, The Netherlands, 2002.

[29] http ://www.exspect.com (retrouvé le 15 août 2009)

[30] Fox, J., Alabassi, A., Patkar, V., Rose, T., Black, E. : *An ontological approach to modelling tasks and goals*, Computers in Biology and Medicine 36 (2006), 837-856.

[31] D. Harel. *Statecharts : A Visual Formalism for Complex Systems*. Science of Computer Programming, 1987.

[32] M. Havey, *Essential Business Process Modeling*, O'Reilly Media, Inc., 2005

[33] W. J. Heaven and A. Finkelstein, "A UML Profile to Support Requirements Engineering with KAOS," IEE Proceedings - Software, 2003.[17] T. Kelly, "Arguing Safety – A Systematic Approach to Managing Safety Cases," inDepartment of Computer Science.

[34] D. Hollingsworth, *The Workflow Reference Model v 1.1,* WFMC-TC-1003, 1995.

[35] ITU-T Recommendation Z.120 Message Sequence Chart (MSC) : http ://www.itu.int/rec/T-REC-Z.120-200404-I/en (retrouvé le 15 août 2009)

[36] Jennings N. R. ; Norman T. J. ; Faratin P. ; O'Brien P. ; Odgers B. ; Autonomous Agents for Business Process Management ; Applied Artificial Intelligence, Volume 14, Number 2, 1 February 2000 , pp. 145-189(45)

[37] V. Kavakli, P. Loucopoulos, Goal-driven business process analysis – application in electricity deregulation, in : Proceedings of CAISE'98, 1998

[38] M. Peeters, A. Zlotta, F. Roucoux, J. De Greve, S. Van Belle, M. Haelterman, D. Ramaekers and G. Dargent. *National Clinical Practice Guidelines of the College of Oncology : A - General framework for a Multidisciplinary Handbook for Oncology. B - Scientific base for clinical pathways for the diagnosis and treatment of colorectal cancer and testicular cancer :* http ://kce.fgov.be/index_en.aspx ?SGREF=5234&CREF=9307 (retrouvé le 3 août 2009)

[39] David Kitchin, Adrian Quark, William Cook and Jayadev Misra, The Orc Programming Language, Proceedings of FMOODS/FORTE, Springer Verlag, LNCS 5522, pp 1–25, 2009.

[40] Knolmayer, G., Endl, R., Pfahrer, M. : *Modeling Processes and Workflows by Business Rules.* In Aalst, W.v.d., Desel, J., Oberweis, A., eds. : Business Process Management : Models, Techniques, and Empirical Studies, volume 1806 of LNCS, Berlin, Springer-Verlag (2000) 16–29

[41] Koubarakis, M., Plexousakis, D. 1999. *Business process modelling and design – a formal model and methodology* BT Technol. J. Vol. 17, No. 4.

[42] R. Kowalski, M. J. Sergot. *A logic-based calculus of events.* New Generation Computing, 4(1) : 67-95, 1986.

[43] R. A. Kowalski, F. Sadri, *Reconciling the Event Calculus with the Situation Calculus*, Journal of Logic Programming, Special Issue on Reasoning about Action and Change, vol. 31, pp. 39-58, 1997.

[44] Liverpool Care Pathway for the Dying Patient (LCP) : http ://www.endoflifecareforadults.nhs.uk/eolc/lcp.htm (retrouvé le 19 août 2009)

[45] Leape LL. preventing medical accidents : is "systems analysis" the answer? Am J Law Med. 2001 ;27(2-3) :145-8.

[46] V. Lifschitz. *Circumscription.* In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, The Handbook of Logic in AI and Logic Programming, volume 3, pages 298–352. Oxford University Press, 1994.

[47] R. Lu, S. Sadiq, A Survey on Comparative Modelling Approaches., in : Proc. BIS'07, 2007, pp. 82–104.

[48] Patient Safety – Making it Happen! Luxembourg Declaration on Patient Safety, European Commission DG Health and Consumer Protection, 2005 : http ://ec.europa.eu/health/ph_overview/Documents/ev_20050405_rd01_en.pdf (retrouvé le 12 août 2009)

[49] Adrian Roberts Sue Middleton (Eds). *Integrated Care Pathways : A Practical Approach to Implementation* Butterworth-Heinemann, 2000.

[50] K. Miller, and W. MacCaull, Modeling and Automated Verification of Healthcare Processes, Centre for Logic and Information Technical Report, CLI-TR 1-2008

[51] R. Milner. The Polyadic Pi-Calculus : A Tutorial, in F.L. Bauer, W. Brauer, H. Schwichtenberg (editors), Logic and Algebra of Specification, Berlin, Springer, 1993.

[52] M. Muehlen, *Workflow-based Process Controlling. Foundation, Design, and Implementation of Workflow-driven Process* Information Systems. Volume 6 of Advances in Information Systems and Management Science. Logos, Berlin (2004).

[53] E. T. Mueller. *Story understanding through multi-representation model construction.* In Graeme Hirst & Sergei Nirenburg (Eds.), Proceedings of the HLT-NAACL 2003 Workshop (pp. 46-53).

[54] Mueller, E.T. : Event Calculus Reasoning through Satisfiability. J. Logic and Computation, Vol. 14, No. 5. Oxford University Press (2004) 703–730

[55] E. T. Mueller, Commonsense Reasoning, Morgan Kaufmann, 2006.

[56] Mulyar N, van der Aalst WM, Peleg M. A Pattern-based Analysis of Clinical Computer-Interpretable Guideline Modeling Languages. J Am Med Inform Assoc. 2007 Aug 21 ;

[57] Mylopoulos, J., Chung, L., Nixon, B., *"Representing and Using Nonfunctional Requirements : A Process-Oriented Approach"*, IEEE Trans. on Sofware. Engineering, Vol. 18 No. 6, June 1992, pp. 483-497.

[58] http ://openclinical.org/gmmintro.html (retrouvé le 12 août 2009)

[59] Verification of Computation Orchestration via Timed Automata : http ://nt-appn.comp.nus.edu.sg/fm/orc (retrouvé le 12 août 2009)

[60] Unified Modeling Language (UML), Version 2.2, OMG, 2009 : http ://www.omg.org/cgi-bin/doc ?formal/09-02-02 (retrouvé le 15 août 2009)

[61] E. Oren and A. Haller. *Formal frameworks for workflow modelling*. Technical Report DERI-TR-2005-04-07, Digital Enterprise Research Institute (DERI), 2005.

[62] Leon J. Osterweil, George S. Avrunin, Bin Chen, Lori A. Clarke, Rachel Cobleigh, Elizabeth A. Henneman, Philip L. Henneman, *Engineering Medical Processes to Improve Their Safety : An Experience Report*, Proceedings IFIP Working Group 8.1 Working Conference on Situational Method Engineering : Fundamentals and Experiences, J. Ralyte, S. Brinkkemper, and B. Henderson-Sellers, eds., Springer, Sept. 2007, pp. 267-282.

[63] L. Padgham, M. Winikoff , *Developing Intelligent Agent Systems : A Practical Guide*, Wiley, 2004

[64] N. Paton and O. Diaz. *Active Database Systems*. ACM Computing Surveys, 31(1) :63-103, 1999.

[65] Peleg M, Tu S, Bury J, Ciccarese P, Fox J, Greenes RA, Hall R, Johnson PD, Jones N, Kumar A, Miksch S, Quaglini S, Seyfang A, Shortliffe EH, Stefanelli M. *Comparing computer-interpretable guideline models : a case-study approach.* J Am Med Inform Assoc 2003 ;10(1) :52—68.

[66] M. Peleg. Guideline and Workflow Models. In : Medical Decision-Making : Computational Approaches to Achieving Healthcare Quality and Safety, Robert A. Greenes (ed.), Elsevier/Academic Press, 2006.

[67] C.A. Petri. Kommunikation mit Automaten. PhD thesis, Institut für instrumentelle Mathematik, Bonn, 1962.

[68] A. Pnueli. *The temporal logic of programs.* In Proceedings of the 18th IEEE Symposium on Foundations of Computer Science, pp. 46–67. 1977.

[69] A. Prior, Time and Modality, Oxford University Press, 1957.

[70] http ://prom.win.tue.nl/research/wiki/prom/start (retrouvé le 15 août 2009)

[71] F. Puhlmann, M. Weske, *Using the Pi-Calculus for Formalizing Workflow Patterns*, in : Proc. BPM'05, 2005, pp. 153–168.

[72] S. B. Ransom, M. S. Joshi, D. Nash, The Healthcare Quality Book : Vision, Strategy, and Tools, Health Administration Press, 2004

[73] M. Reichert and P. Dadam. ADEPTflex : Supporting Dynamic Changes of Workflow without Loosing Control. Journal of Intelligent Information Systems, 10(2) :93–129, 1998.

[74] R. Reiter. *Knowledge in Action : Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, 2001

[75] SB Rinderle ; Schema Evolution in Process Management Systems ; Dissertation zur Erlangung des Doktorgrades ; informatik.uni-ulm.de ; 2004

[76] Rubin, V., Günther, C., van der Aalst, W., Kindler, E., Van Dongen, B. & Schäfer, W. (2007) Process Mining Framework for Software Processes. Software Process Dynamics and Agility, LNCS, 4470/2007, 169-181, Springer-Verlag, Berlin.

[77] N. Russell, A. ter Hofstede, and W. van der Aalst. *newYAWL : specifying a workflow reference language using coloured Petri nets.* In Proceedings of the 8th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, CPN'07, 2007.

[78] M. P. Shanahan, *Perception as Abduction : Turning Sensor Data into Meaningful Representation*, Cognitive Science, vol 29 (2005), pages 103-134.

[79] Abductive Event Calculus Planners : http ://www.iis.ee.ic.ac.uk/~mpsha/planners.html (retrouvé le 19 août 2009)

[80] Sriram Srinivasan, Alan Mycroft, Kilim : *Isolation-Typed Actors for Java*, European Conference on Object Oriented Programming ECOOP 2008, Cyprus.

[81] Sharon E. Straus, W. Scott Richardson, Paul Glasziou, R. Brian Haynes, Sharon E. Strauss ; *Evidence Based Medicine* ; Churchill Livingstone ; 3rd edition, 2005.

[82] D.R. Sutton and J. Fox, 'The syntax and semantics of the proforma guideline modeling language', JAMIA,ed American Medical Informat- ics Association, 10(5), 433–443, (2003).

[83] http ://www.fr.tibco.com/multimedia/ds-iprocess-conductor_tcm18-782.pdf (retrouvé le 3 août 2009)

[84] http ://www.healthgrades.com/media/dms/pdf/PatientSafetyInAmericanHospitalsStudy2006.pdf

[85] Van den Heede, K., Sermeus, W., Diya, L., Lesaffre, E., Vleugels, A. (2006). Adverse outcomes in Belgian acute hospitals : retrospective analysis of the national hospital discharge dataset. International Journal for Quality in Healthcare.

[86] van der Aalst. W. M. P. : *Three Good reasons for Using a Petri-net-based Workflow Management System.* In proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96), Cambridge, Massachusetts (1996)

[87] W.M.P. van der Aalst. *The Application of Petri Nets to Workflow Management.* The Journal of Circuits, Systems and Computers, 8(1) :21-66, 1998.

[88] W. M. P. van der Aalst, A.P. Barros, A.H.M. ter Hofstede, and B. Kiepuszewski. *Advanced workflow patterns.* In O. Etzion and P. Scheuermann, editors, Proceedings of the Fifth IFCIS International Conference on Cooperative Information Systems (CoopIS 2000), volume 1901 of LNCS, pages 18-26, Eilat, Israel, 2000. Springer.

[89] W.M.P. van der Aalst and A.H.M. ter Hofstede. *YAWL : Yet Another Workflow Language*, QUT Technical report, FIT-TR-2002-06, Queensland University of Technology, Brisbane, 2002.

[90] W.M.P van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. *Workflow Patterns.* Distributed and Parallel Databases, 14(3), pages 5-51, July 2003

[91] van der Aalst, W.M.P. : *Pi calculus versus petri nets : Let us eat "humble pie" rather than further inflate the "pi hype"*, 2005. http ://is.tm.tue.nl/research/patterns/ download/pi- hype.pdf (retrouvé le 3 août 2009)

[92] WMP van der Aalst, M Weske, D Grünbauer - Case handling : a new paradigm for business process support - Data & Knowledge Engineering, 2005 – Elsevier.

[93] W.M.P. van der Aalst and A.H.M. ter Hofstede. *YAWL : Yet Another Workflow Language. Information Systems* , 30(4) :245-275, 2005.

[94] K. Vanhaecht & W. Sermeus, Scénario pour l'élaboration d'un itinéraire clinique, sa mise en oeuvre et son évaluation. Plan en 30 étapes du "Réseau Itinéraires Cliniques", Centrum voor Ziekenhuis- en Verplegingswetenschap, K.U.L. Leuven, 2003

[95] K. Vanhaecht, M. Bollmann, K. Bower et al., *Prevalence and use of clinical pathways in 23 countries — an international survey by the European Pathway Association.* Journal of Integrated Care Pathways, 2006, 10, 28-34.

[96] Axel van Lamsweerde, *Requirements Engineering : from system goals to UML models to software specifications*, Wiley, 2009.

[97] Van Roy Peter, Haridi Seif, "Concepts, Techniques, and Models of Computer Programming", MIT Press, 2004, p. 60.

[98] Ian Wehrman, David Kitchin, William R. Cook. Jayadev Misra, *A Timed semantics of Orc*, Theoretical Computer Science, August 2008.

[99] Workflow Management Coalition, Terminology & Glossary, Document Number WFMC-TC-1011, Workflow Management Coalition, 1999

[100] S. A. White, "Process Modeling Notations and Workflow Patterns" : http ://www.bpmn.org/Documents/Notations%20and%20Workflow%20Patterns.pdf (retrouvé le 3 août 2009)

[101] S. A. White, *Specification of BPMN v1.0*, for Business Process Management Initiative (BPMI), 2004 : http ://www.bpmn.org/Documents/BPMN%20V1-0%20May%203%202004.pdf (retrouvé le 12 août 2009)

[102] http ://www.whitestein.com/go-bpmn (retrouvé le 3 août 2009)

[103] P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and N. Russell. Pattern- Based Analysis of the Control-Flow Perspective of UML Activity Diagrams. In Proc. of 24th Int. Conf. on Conceptual Modeling (ER05), volume 3716 of LNCS, pages 63–78. Springer Verlag, 2005.

[104] M. Wooldridge, N. R. Jennings, and D. Kinny. *The Gaia Methodology for Agent-Oriented Analysis and Design.* In Journal of Autonomous Agents and Multi-Agent Systems. 3(3) :285-312. 2000.

[105] P. Yolum and M. P. Singh. *Reasoning About Commitments in the Event Calculus : An Approach for Specifying and Executing Protocols.* Annals of Mathematics and Artificial Intelligence (AMAI), Special Issue on Computational Logic in Multi-Agent Systems, 2003.

[106] Zander, K. & Bower, K. Implementing systems for managing care. The Center for Case Management, 2000 Boston.

Annexe A : Rappel de logique

Cette annexe présente les notions élémentaires de logique nécessaires pour appréhender le calcul événementiel basé sur la logique du premier ordre multi-typée avec égalité. Dans ce travail, la logique événementielle est utilisée pour représenter à la fois les aspects opérationnels et intentionnels de processus de soins.

La logique propositionnelle

La logique propositionnelle ou logique des propositions est le langage logique le plus simple. Il se définit par une syntaxe, une sémantique et un système de déduction appelé théorie de la preuve qui permet d'effectuer des raisonnements.

Syntaxe

La syntaxe est un ensemble de règles qui définit les expressions bien formées d'un langage. Le langage de la logique des propositions est composé :

- d'un ensemble de propositions
- des connecteurs logiques : $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
- des symboles de ponctuation : '(' et ')'

Deux règles récursives suffisent à définir complètement la syntaxe de la logique des proposition (adapté de van Lamsweerde [96]). Les voici dans une représentation de type BNF [8] :

<propositionAtomique> : := vrai | faux | <symbolePropositionnel>
<formule> : := <propositionAtomique> | (¬<formule>)
| (<formule> ∧ <formule>) | (<formule> ∨ <formule>)
| (<formule> ⇒ <formule>) | (<formule> ⇔ <formule>)

Par exemple :

patientDiabétique et *glucoseElevé* sont des symboles propositionnels.

glucoseElevé ⇒ *patientDiabétique* est une formule propositionnelle.

Il est possible dans certains cas d'omettre les parenthèses en appliquant les règles de précédence des opérateurs : $\neg > \wedge > \vee > \Rightarrow > \Leftrightarrow$
Dans le cadre de la syntaxe, aucune valeur de vérité n'est attachée aux proposition. C'est le rôle de la sémantique du langage de définir leur valeur.

Sémantique

La sémantique donne les règles qui permettent de définir la valeur de vérité d'une formule propositionnelle (ou proposition). Cette valeur de vérité est indépendante de la signification attribuée aux symboles propositionnels qui composent cette formule. La logique propositionnelle est une logique à deux valeurs. Chaque proposition peut-être vraie (V) ou fausse (F).

val_I est une fonction qui assigne une valeur de vérité à chaque symbole propositionnel d'une proposition sujette à l'interprétation I.

> $val_I : S_p \mapsto \{V, F\}$ ou S est l'ensemble des symboles propositionnels d'une proposition p sujette à l'interprétation I.

Une interprétation I d'un ensemble de propositions logiques est une fonction totale qui assigne une valeur de vérité à chacune des propositions sujettes à cette interprétation.

> $I : P_I \mapsto \{V, F\}$ ou P_I est l'ensemble des propositions logiques sujettes à l'interprétation I.

Les tableaux suivants (tables de vérité) décrivent la sémantique de la logique propositionnelle :

$I(vrai)$	$I(faux)$	$I(p)$
V	F	$val_I(p)$

$I(p)$	$I(\neg p)$
V	F
F	V

$I(p_1)$	$I(p_2)$	$I(p_1 \wedge p_2)$	$I(p_1 \vee p_2)$	$I(p_1 \Rightarrow p_2)$	$I(p_1 \Leftrightarrow p_2)$
V	V	V	V	V	V
V	F	F	V	F	F
F	V	F	V	V	F
F	F	F	F	V	V

où p, p_1 et p_2 sont des propositions logiques quelconques

Par exemple, soit la proposition :

> $glucoseElevé \Rightarrow patientDiabétique$

et une interprétation I_1 dans laquelle les valeurs de vérités suivantes sont assignées aux symboles propositionnels :

$val_I(glucoseElevé) = V$
$val_I(patientDiabétique) = V$

la valeur de vérité de la proposition dans cette interprétation est :

$$I_1(glucoseElevé \Rightarrow patientDiabétique) = V$$

pour une autre inteprétation I_2 dans laquelle :

$val_I(glucoseElevé) = V$
$val_I(patientDiabétique) = F$

la valeur de vérité dans cette autre interprétation est :

$$I_2(glucoseElevé \Rightarrow patientDiabétique) = F$$

L'expression $I \models p$ signifie que l'interprétation I rend la proposition p vraie. On dit que I est un modèle de p. Deux formules p et q sont sémantiquement équivalentes si $p \models q$ et $q \models p$ ce qui se note $p \equiv q$.

La notion de modèle peut-être étendue à un ensemble de formules. Une formule p est la conséquence logique d'un ensemble F de formules si tout modèle de F est également un modèle de p.

Théorie de la preuve

On appelle théorie de la preuve un ensemble de règles d'inférences qui à partir d'un ensemble donné d'expressions permettent d'en dériver de nouvelles. Pour chaque règle, les propositions données en entrée sont appelées prémices et les propositions dérivées, conclusions. Il existe une relation de dérivabilité entre un ensemble de formules F et une formule p si l'on peut dériver p à partir des formules de F en appliquent seulement des axiomes et des règles d'inférence. Cette relation se note $F \vdash p$.

Une règle bien fondée garantit que les conclusions sont vraies dans toutes les interprétations qui rendent les prémices vrais : si $F \vdash p$ alors $F \models p$. Un règle complète garantit que si $F \models p$ alors $F \vdash p$.

Le tableau suivant illustre certaines des principales règles de dérivation :

Nom de la règle	Prémices	Conclusions
Chaînage	$p_1 \Rightarrow p_2, p_2 \Rightarrow p_3$	$p_1 \Rightarrow p_3$
Modus ponens	$p_1 \Rightarrow p_2, p_1$	p_2
Résolution	$p_1 \vee p_2, \neg p_1 \vee p_3$	$p_2 \vee p_3$

où p_1, p_2 et p_3 sont des propositions logiques quelconques

La logique des prédicats ou logique du premier ordre

La logique du premier ordre est une extension de la logique des propositions. Elle augmente son expressivité en introduisant :

- des variables
- des constantes
- des termes
- des quantifications sur les variables

On utilise les termes pour représenter des objets du domaine et des atomes (prédicats atomiques) pour représenter les relations qui existent entre les objets du domaine.

Syntaxe

Les formules BNF suivantes définissent la syntaxe de la logique du premier ordre (adapté de van Lamsweerde [96]) :

<terme> : := <constante> | <variable> | <symboleFonctionnel>(<terme>*)
<atome> : := vrai | faux | <symboleDePrédicat>(<terme>*)
<formule> : := <atome> | (¬<formule>
| (<formule> ∧ <formule>) | (<formule> ∨ <formule>)
| (<formule> ⇒ <formule>) | (<formule> ⇔ <formule>)
| (∀ <variable> <formule>) | (∃ <variable> <formule>)

La portée d'un quantificateur $\exists v$ dans la formule $\exists v\, f$ est f. Il en va de même pour le quantificateur $\forall$. L'occurrence d'un variable v dans une formule qui est à portée d'un quantificateur est dite liée. Autrement, la variable est dite libre. Une phrase est une formule qui ne contient pas de variables libres. Dans ce travail, les formules sont toutes des phrases. Les variables qui ne font pas l'objet d'une quantification explicite sont implicitement quantifiée universellement.

$Traité(a_1, a_2, a_3)$ est un symbole de prédicat d'arité 3 (trois arguments).

$TensionArtérielleSystolique(a)$ est un symbole de fonction d'arité 1 dont la valeur de retour est la tension artérielle systolique du patient passé en paramètre.

$Dupont$, $Infirmier$, $Contramal$ sont des constantes qui expriment des objets du domaines (respectivement, une personne, un rôle et un médicament).

v est une variable.

$Tension(Dupont)$ est un terme.

$Traité(Dupont, Infirmier, Contramal)$ est un atome (ou prédicat atomique) et également une phrase : le patient Dupont a été traité par un infirmier avec du Contramal.

$\forall p\,(GlucoseElevé(p) \Rightarrow Diabétique(p))$ est une phrase : tout patient dont la glycémie est élevée est diabétique (il s'agit d'une simplification de la réalité médicale).

Sémantique

Lorsqu'on veut définir l'interprétation I d'un ensemble de prédicats, il faut également définir le domaine considéré. Le domaine correspond à l'ensemble des objets représentés par des termes.

En logique du premier ordre, la fonction val_I a les significations suivante en fonction de l'élément sur lequel elle porte :

- pour une constante c, $val_I(c)$ désigne l'élément du domaine correspondant.
- pour une variable non-quantifiée v, $val_I(v)$ désigne l'élément du domaine correspondant.
- pour un symbole de fonction f, $val_I(f)$ désigne la fonction sur le domaine correspondante.
- pour un symbole de prédicat p d'arité n, $val_I(p)$ désigne la relation n-aire sur le domaine correspondante.

Les règles suivantes ainsi que les règles sémantiques de la logique des propositions définissent récursivement la sémantique de la logique du premier ordre pour une certaine interprétation I [96] :

$I(c) = val_I(c)$ si c est une constante

$I(v) = val_I(v)$ si v est une variable non quantifiée

$I(f(t_1, \ldots, t_n)) = (val_I(f))(I(t_1, \ldots, I(t_n))$ si f est un symbole de fonction sur les termes t_i

$I(vrai) = V$

$I(faux) = F$

$I(P(t_1, \ldots, t_n)) = (val_I(P))(I(t_1, \ldots, I(t_n))$ si P est un symbole de prédicat sur les termes t_i

$I((\forall v)P) = V$ si $I_{\{v \leftarrow e\}}(P) = V$ pour chaque élément e du domaine

$I((\forall v)P) = F$ si $I_{\{v \leftarrow e\}}(P) = F$ pour certains éléments e du domaine

$I((\exists v)P) = V$ si $I_{\{v \leftarrow e\}}(P) = V$ pour certains éléments e du domaine

$I((\exists v)P) = F$ si $I_{\{v \leftarrow e\}}(P) = F$ pour chaque élément e du domaine

P représente une formule de prédicats quelconque et $I_{\{v \leftarrow e\}}$ représente l'interprétation qui étend l'interprétation I lorsque la variable v représente l'élément du domaine e.

Théorie de la preuve

Les formules de la théorie de la preuve de la logique du premier ordre incluent celles de la logique des propositions et d'autres qui permettent l'inférence de nouvelles formules.

L'instanciation permet qu'une formule comportant un quantificateur universel soit instanciée par rapport à un quelconque terme t. Le quantificateur est ainsi éliminé et toutes les occurrences de la variable quantifiée sont remplacées par ce terme :

$\forall v\,(P)$ devient par instanciation de v en t : $P_{[x/t]}$ (signifie le prédicat P dans lequel toutes les occurences de v ont été remplacées par t).

La substitutivité fonctionnelle et des prédicats s'appuient sur le symbole de prédicat d'égalité dont '=' est la représentation infixe. Si t_i et u_i sont des termes, f un symbole de fonction et P un symbole de prédicat :

$t_1 = u_1, \ldots, t_n = u_n$ permet d'écrire $f(t_1, \ldots, t_n) = f(u_1, \ldots, u_n)$
$t_1 = u_1, \ldots, t_n = u_n$ permet d'écrire $P(t_1, \ldots, t_n) \Leftrightarrow P(u_1, \ldots, u_n)$

La logique du premier ordre multi-typée

Le concept de type (également appelé "sort" en anglais) est important pour certaines extensions de la logique du premier ordre dont fait partie le calcul événementiel [55]. Tout comme la notion de type dans les langages de programmation, ce concept permet de paritionner les objets du domaine en différentes sous-classes. Sémantiquement, c'est le domaine qui est partitionné en sous-domaines disjoints, un sous-domaine par type. Chaque variable peut prendre des valeurs dans le sous-domaine qui correspond à son type. Les prédicats et les fonctions doivent être restreints pour n'accepter que des arguments de types définis. De la même façon, les fonctions sont contraintes pour ne retourner que des valeurs d'un certain type.

Les types constituent une extension utile à la logique du premier ordre sans modifier fondamentalement ses propriétés. Néanmoins, la notion d'interprétation doit être étendu pour supporter les types.

Les types se présentent sous la forme d'annotations sur les variables quantifiées. Voici une phrase dont la variable p est non-typée :

> $\forall p\,(Patient(p) \wedge CRPElevée(p) \Rightarrow SyndromeInflammatoire(p))$ signifie que tout objet du domaine qui est un patient et dont la CRP est élevée présente un syndrome inflammatoire.

Et voici la même avec une annotation de type :

> $\forall p : Patient\,(CRPElevée(p) \Rightarrow SyndromeInflammatoire(p))$ signifie que tout patient dont la CRP est élevée présente un syndrome inflammatoire.

Pour chaque type, il peut exister un nombre à priori infini de variables de ce type. La notion de sous-type (sous-endemble d'un ensemble de types) est également supportée.

exemple de type : $Personne$, $Patient$, $Salle$. Le type $Patient$ est un sous-type de $Personne$. Le type de la constante $Dupont$ est $Patient$. Le type des constantes $Chambre$ et $SalleDOpération$ est salle.

La logique du premier ordre avec l'égalité

L'égalité est le prédicat binaire '=' typiquement écrit sous une forme infixe. Une équation est une formule de la forme $t = u$ où t et u sont des termes. L'égalité est donc un symbole relationnel ordinaire. Néanmoins pour concéder à ce symbole de prédicat les propriétés arithmétiques habituelles de symétrie, transitivité, reflexivité, ..., il faut adjoindre au système déductif une série d'axiomes dits axiomes de l'égalité.